

This user guide describes the features and behavior of the ALTPLL\_RECONFIG megafunction that you can configure through the parameter editor in the Quartus® II software.



This user guide assumes that you are familiar with megafunctions and how to create them. If you are unfamiliar with Altera megafunctions or the parameter editor, refer to the *Introduction to Megafunctions User Guide*.

Phase-locked loops (PLLs) use divide counters and voltage-controlled oscillator (VCO) phase taps to perform frequency synthesis and phase shifts. In enhanced and fast PLLs, you can reconfigure the counter settings as well as phase shift the PLL output clock in real time. You can also change the charge-pump and loop-filter components, which dynamically affect the PLL bandwidth. The ALTPLL\_RECONFIG megafunction implements reconfiguration logic to facilitate dynamic real-time reconfiguration of PLLs in Altera devices. You can use the megafunction to update the output clock frequency, PLL bandwidth, and phase shifts in real time, without reconfiguring the entire FPGA.

## Features

The ALTPLL\_RECONFIG megafunction offers the following additional features to the ALTPLL megafunction:

- Reconfiguration of pre-scale counter ( $N$ ) parameters.
- Reconfiguration of feedback counter ( $M$ ) parameters.
- Reconfiguration of post-scale output counter ( $C$ ) parameters.
- Reconfiguration of delay element or phase shift of each counter. For Stratix® III, Stratix IV, Cyclone® III, Cyclone IV, HardCopy® III, HardCopy IV, and Arria® II GX devices, use the ALTPLL megafunction to access this feature.
- Dynamic adjustment of the charge-pump current and loop-filter components to facilitate dynamic reconfiguration of the PLL bandwidth. This feature is available only in Arria GX, HardCopy II, Stratix II, Stratix II GX, Stratix III, and Stratix IV devices.
- Reconfiguration from multiple configuration files using external read-only memory (ROM) in user mode. This feature is available only in Stratix III, Stratix IV, Cyclone III, Cyclone IV, and Arria II GX devices. The ALTPLL\_RECONFIG supports reconfiguration from Memory Initialization File (.mif) and Hexadecimal File (.hex).



For more details about these features, refer to the *Clock Networks and PLLs* chapter of the respective device handbook.

## Common Applications

Use the ALTPLL\_RECONFIG megafunction in designs that must support dynamic changes in the frequency and phase shift of clocks and other frequency signals. The megafunction is also useful in prototyping environments because it allows you to sweep PLL output frequencies and dynamically adjust the output clock phase. For example, a system generating test patterns is required to generate and transmit patterns at 50 or 100 MHz, depending on the device under test. Reconfiguring the PLL components in real-time allows you to switch between two such output frequencies within a few microseconds. You can also adjust the clock-to-output (tCO) delays in real-time by changing the output clock phase shift. This approach eliminates the need to regenerate a configuration file with the new PLL settings.

Reconfigurable PLLs are very useful in DDR 2 and DDR 3 interfaces to implement the dynamic data path (via the ALTMEMPHY megafunction). The PLL is needed to drive the DLL used in the dynamic external memory interface operation. This operation requires dynamic phase-shifting.



For more information about dynamic phase-shifting in DDR 2 and DDR 3 interfaces, refer to the [ALTMEMPHY Megafunction User Guide](#).

In addition, you can dynamically configure Stratix III, Stratix IV, Cyclone III, Cyclone IV, and Arria II GX PLLs by using multiple configuration files stored on the external ROM.

## Device Family Support

The megafunction supports the Stratix series (excluding Stratix V), HardCopy series, Arria GX series, and Cyclone series devices.

## Resource Utilization and Performance

For details about the resource usage and performance of the ALTPLL\_RECONFIG megafunction in various devices, refer to the compilation reports in the Quartus II software.

To view the compilation reports for the ALTPLL\_RECONFIG megafunction in the Quartus II software, follow these steps:

1. On the Processing menu, click **Start Compilation** to run a full compilation.
2. After compiling the design, on the Processing menu, click **Compilation Report**.
3. In the Table of Contents browser, expand the **Fitter** folder by clicking the “+” icon.
4. Under **Fitter**, expand **Resource section**, and select **Resource Usage Summary** to view the resource usage information.
5. Under **Fitter**, expand **Resource section**, and select **Resource Utilization by Entity** to view the resource utilization information.

## Parameter Settings

Altera recommends that you configure the megafunction using the MegaWizard™ Plug-In Manager. This section describes the parameters in the ALTPLL\_RECONFIG parameter editor.

Expert users may choose to instantiate and configure the megafunction using the clear box generator.

[Table 1](#) lists the parameter settings for the ALTPLL\_RECONFIG megafunction.

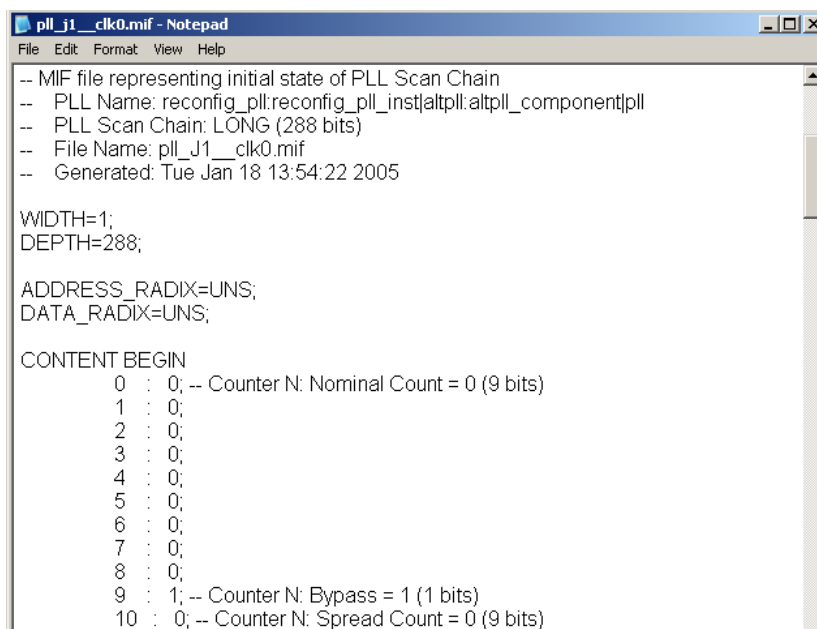
**Table 1. ALTPLL\_RECONFIG Parameter Settings**

Page	Options	Description
Parameter Settings	Currently Selected Device Family	Specifies the chosen device family.
	Which scan chain type will you be using?	Scan chain is serial shift register chain that is used to store settings. It acts like a cache. When you assert the reconfig signal, the PLL is reconfigured with the values in the cache. The type of scan chain must follow the type of PLL to be reconfigured. For Arria GX, Stratix II, Stratix II GX, and HardCopy II devices—Specifies the scan chain type as either <b>Enhanced</b> or <b>Fast</b> . For Stratix and Stratix GX devices—Specifies the scan chain type as either <b>Long chain</b> or <b>Short chain</b> . For Stratix III, Stratix IV, HardCopy III, and HardCopy IV devices—Specifies the scan chain type as either <b>Top/Bottom</b> or <b>Left/Right</b> . For Cyclone III, Cyclone IV, and Arria II GX devices—The scan chain type has a default value of <b>Left/Right</b> .
	Do you want to specify the initial value of the scan chain?	Specifies the initial value of the scan chain. Select <b>No, leave it blank</b> to not specify a file or select <b>Yes, use this file for the content data</b> to browse for a <b>.hex</b> or <b>.mif</b> file. For Arria GX, Arria II GX, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, Stratix IV, HardCopy II, HardCopy III, and HardCopy IV devices—You can also choose to initialize from ROM by turning on <b>Do not use pre-initialized RAM - initialize from ROM instead</b> . For Cyclone III and Cyclone IV devices—The option to initialize from a ROM is not available. However, you can choose to add ports to write to the scan chain from an external ROM during runtime by turning on <b>Add ports to write to the scan chain from external ROM during run time</b> .
	Add ports to write to the scan chain from external ROM during run time	This option is only available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices. This option takes advantage of cycling multiple configuration files, which are stored in external ROMs during user mode. This capability is demonstrated in the functional description section, <a href="#">“Functional Description—Implementing Multiple Reconfiguration Using an External ROM”</a> on page 6.

**Table 1. ALTPLL\_RECONFIG Parameter Settings**

Page	Options	Description
EDA		Specifies the libraries needed for functional simulation.
	Generate netlist	Specifies whether to turn on the option to generate synthesis area and timing estimation netlist.
Summary		<p>Specifies the types of files to be generated. A gray checkmark indicates a file that is automatically generated; a red checkmark indicates an optional file.</p> <p>Choose from the following types of files:</p> <ul style="list-style-type: none"> <li>■ AHDL Include file (<i>&lt;function name&gt;.inc</i>)</li> <li>■ VHDL component declaration file (<i>&lt;function name&gt;.cmp</i>)</li> <li>■ Quartus II symbol file (<i>&lt;function name&gt;.bsf</i>)</li> <li>■ Instantiation template file (<i>&lt;function name&gt;_inst.v</i> or <i>&lt;function name&gt;_inst.vhd</i>)</li> <li>■ Verilog HDL block box file (<i>&lt;function name&gt;_bb.v</i>)</li> </ul> <p>If <b>Generate netlist</b> option is turned on, the file for that netlist is also available (<i>&lt;function name&gt;_syn.v</i>).</p>

You can open a **.mif** in a text editor to make use of the comments embedded within the file. These comments show you the scan chain values and positions based on your design parameterization (see [Figure 1](#)). If you open a **.mif** in the Quartus II software, you can regenerate the **.mif** in the ALTPLL parameter editor to restore the comments.

**Figure 1. MIF file**


```

pll_j1_clk0.mif - Notepad
File Edit Format View Help

-- MIF file representing initial state of PLL Scan Chain
-- PLL Name: reconfig_pll:reconfig_pll_inst|altpll:altpll_component|pll
-- PLL Scan Chain: LONG (288 bits)
-- File Name: pll_j1_clk0.mif
-- Generated: Tue Jan 18 13:54:22 2005

WIDTH=1;
DEPTH=288;

ADDRESS_RADIX=UNS;
DATA_RADIX=UNS;

CONTENT BEGIN
  0 : 0; -- Counter N: Nominal Count = 0 (9 bits)
  1 : 0;
  2 : 0;
  3 : 0;
  4 : 0;
  5 : 0;
  6 : 0;
  7 : 0;
  8 : 0;
  9 : 1; -- Counter N: Bypass = 1 (1 bits)
  10 : 0; -- Counter N: Spread Count = 0 (9 bits)

```

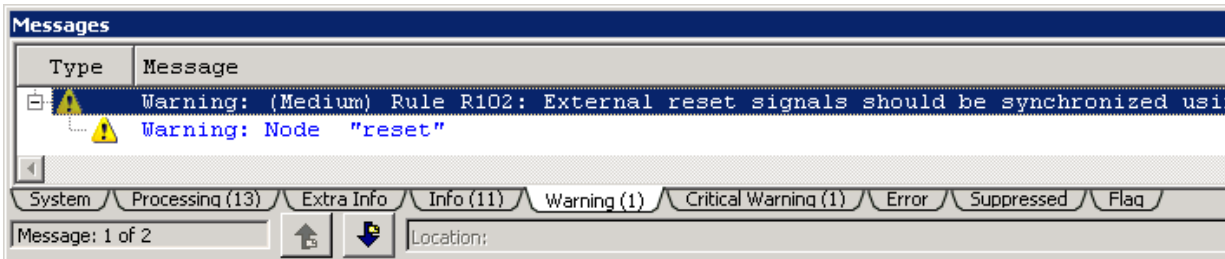


For more information about implementing PLL reconfiguration in the supported Stratix series, refer to [AN 282: Implementing PLL Reconfiguration in Stratix & Stratix GX Devices](#), [AN 367: Implementing PLL Reconfiguration in Stratix II Devices](#) and [AN 454: Implementing PLL Reconfiguration in Stratix III Devices](#).

## Checking Design Violations With the Design Assistant

The Design Assistant is a design rule checking tool that allows you to check for design issues early in the design flow. When you run the Design Assistant in the Quartus II software for the ALTPLL\_RECONFIG megafunction, you might receive the warning message shown in [Figure 2](#).

**Figure 2. Warning Message in Design Assistant**



This message appears because there is a combinational logic in the megafunction that connects the synchronous signal to the asynchronous external reset signal. To fix the issue, you must synchronize the external reset signal outside the megafunction.

To synchronize the external reset signal, use the sample Verilog HDL code shown in [Example 1](#). In the example, the input of sync\_reset\_dff1 is connected to the external reset pin, and the output of sync\_reset\_dff2 is connected to the reset input port of the ALTPLL\_RECONFIG megafunction.

### Example 1. Code to Synchronize External Reset Signal

```
module synch_reg (reset, reconfig_clk, sync_reset_dffe2);
    input reset, reconfig_clk;
    output sync_reset_dffe2;
    reg sync_reset_dff1, sync_reset_dffe2;
    always @(posedge reconfig_clk)
    begin
        sync_reset_dff1 = reset;
    end
    always @(posedge reconfig_clk)
    begin
        sync_reset_dffe2 = sync_reset_dff1;
    end
endmodule
```

## Simulation

You can perform functional and gate-level timing simulations of the megafunction.



For more information, refer to the appropriate chapter in the *Simulation* section in volume 3 of the *Quartus II Handbook*.

If phase-shifting occurs after a PLL reconfiguration, use gate-level timing simulation instead of functional simulation to verify the correct counter settings and phase shifts. For non-zero PLL phase shifts, the frequency of the output clocks after a reconfiguration is correct, but the phase may be incorrect. If the phase shift is significant, use gate-level timing simulation to verify the timing behavior.

## Functional Description—Implementing Multiple Reconfiguration Using an External ROM

The ALTPLL\_RECONFIG megafunction allows you to reconfigure the PLL using an external ROM with multiple configuration files. With this feature, you can perform the following:

- Specify an external ROM and feed its content to the ALTPLL\_RECONFIG megafunction.
- Use the megafunction with multiple PLL configuration settings that are stored in configuration files during user mode.
- Use the megafunction with applications that require flexible dynamic-shifting of PLL settings during user mode.
- Reconfigure the initial PLL settings from a source other than an embedded random-access memory (RAM), such as an off-chip flash device, which is useful in HardCopy-type applications.



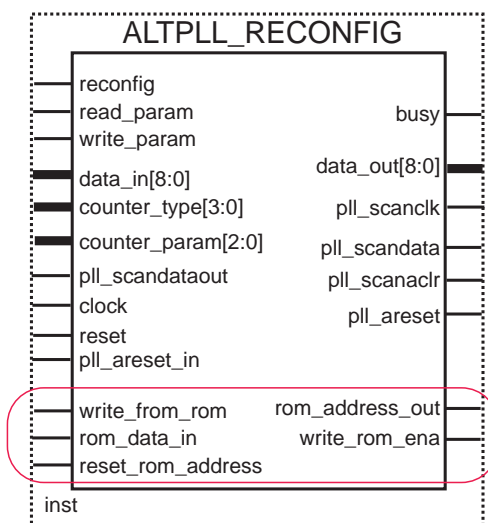
This feature is available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices only.

To support reconfiguration from multiple configuration files, the ALTPLL\_RECONFIG megafunction has three input ports and two output ports:

- The `write_from_rom` input port signals the ALTPLL\_RECONFIG megafunction instantiation to write to the scan cache from the ROM.
- The `rom_data_in` input port holds data from the ROM.
- The `reset_rom_address` input port lets you restart the read process from the ROM. The data arrives serially from the ROM, starting from bit 0.
- The `rom_address_out` output bus holds the current address of the ROM data to be written to the scan cache.
- The `write_rom_ena` output port enables the intended ROM to be read out.

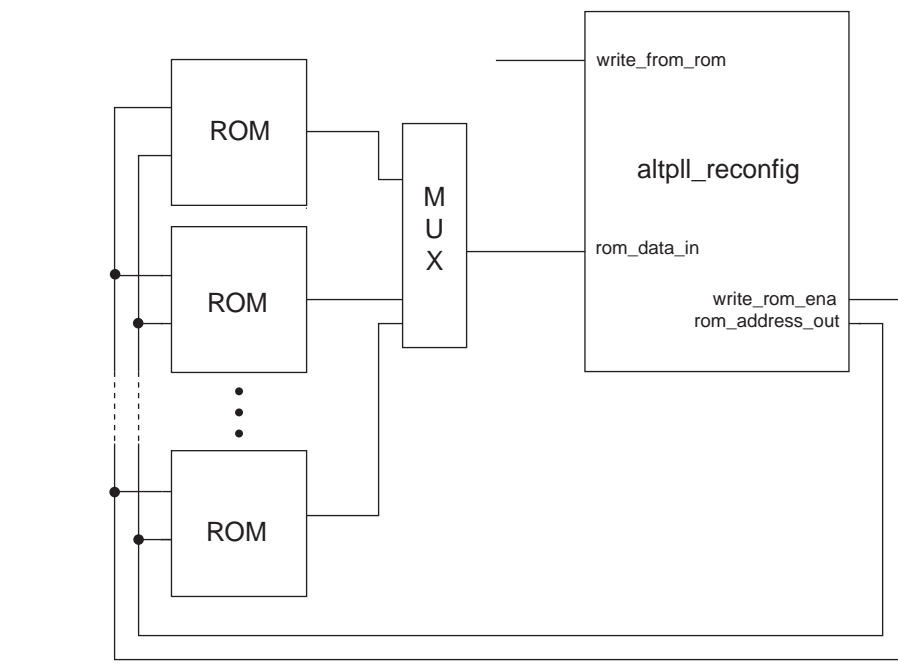
The input and output ports to support reconfiguration using multiple configuration files are shown in [Figure 3](#), circled in red.

**Figure 3. Ports to Support Reconfiguration Using Multiple Configuration Files**



The reconfiguration feature using multiple configuration files allows you to feed data from multiple ROMs to a multiplexer that feeds the `rom_data_in` port. [Figure 4](#) shows a sample design. In this scheme, the `write_rom_ena` signal feeds back to the ROM as the enable signal, which allows the ROM to be read out. The `rom_address_out` bus provides the intended ROM address, which determines the exact ROM data.

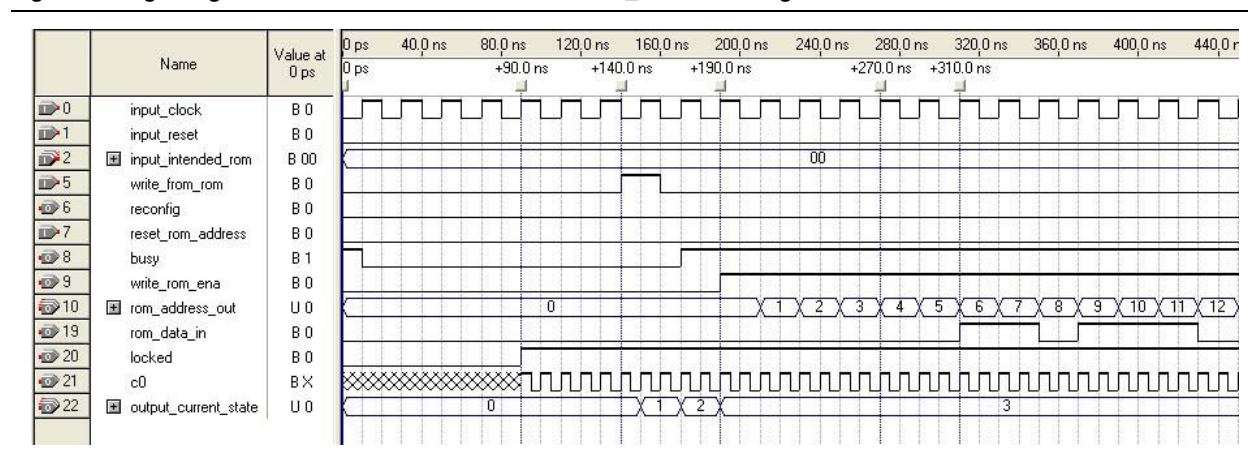
**Figure 4. Typical Scheme for Reconfiguring PLLs from External ROMs**



To copy the data from a ROM to the ALTPLL\_RECONFIG megafunction scan cache (a memory location that stores the PLL reconfiguration settings), you must hold the `write_from_rom` signal high for 1 clock cycle. The megafunction asserts the busy signal on the first rising edge of the clock after the `write_from_rom` signal goes high. The busy signal remains asserted until all the bits are written into the scan cache.

On the second rising edge of the clock after the `write_from_rom` signal goes low again, the intended ROM address for the write operation appears on the `rom_address_out` port. The data of the ROM specified by the intended address on `rom_address_out` is fed to the `rom_data_in` input port of the ALTPLL\_RECONFIG megafunction instantiation. The `write_rom_ena` signal is also asserted on the second rising edge of the clock after the `write_from_rom` signal goes low again (refer to Figure 5).

**Figure 5. Beginning Write to the Scan Cache of the ALTPLL\_RECONFIG Megafunction from the ROM**

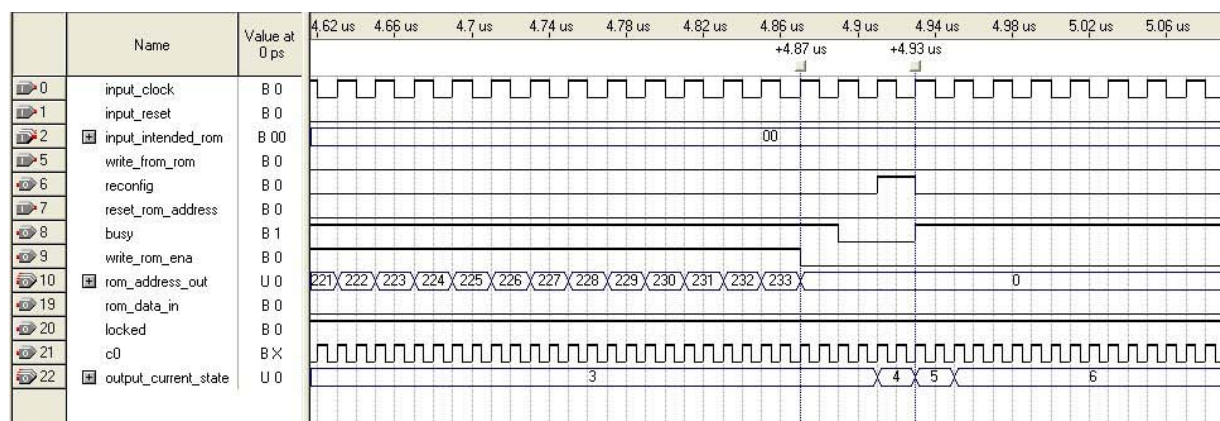


The writing-to-scan cache process continues until the address reaches the specific size of the scan cache (234 for Stratix III device top and bottom PLL, 180 for Stratix III device left and right PLL, and 144 for Cyclone III PLL). This process is completed when the busy signal is deasserted. This means that the scan cache of the ALTPLL\_RECONFIG megafunction is written with the intended reconfiguration settings from the ROM.



After this, the reconfig signal can be asserted for 1 clock cycle to reconfigure the PLL to the intended settings that have been written to the scan cache of the ALTPLL\_RECONFIG megafunction (refer to [Figure 6](#)).

**Figure 6. Completing Write to the Scan Cache of the ALTPLL\_RECONFIG Megafunction from the ROM <sup>(1)</sup>**

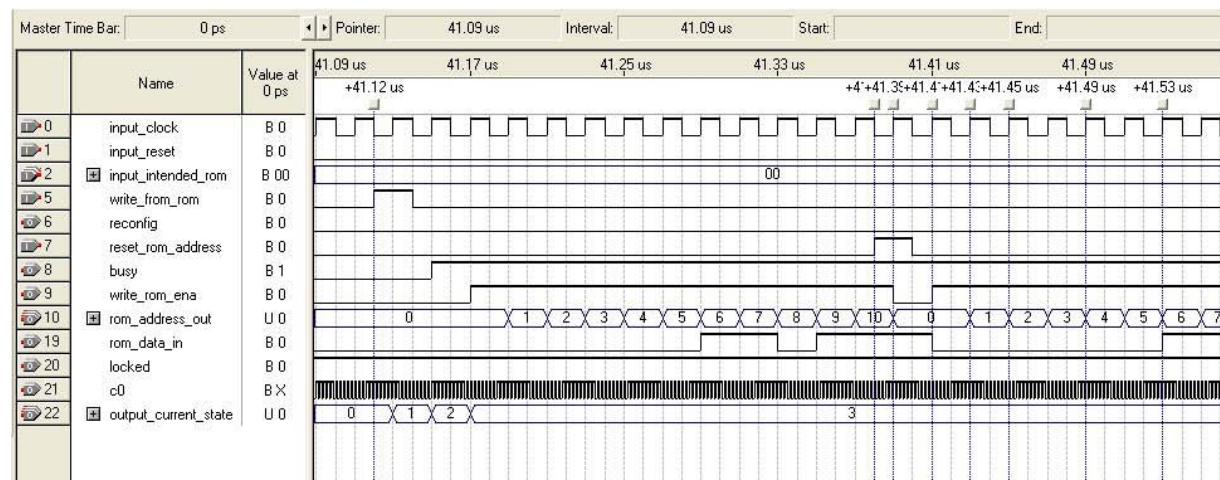


**Note to Figure 6:**

(1) This figure also shows the beginning of the reconfiguration process.

If you assert the reset\_rom\_address signal, the write\_rom\_ena signal is deasserted for 1 clock cycle and the rom\_address\_out signal resets. When the write\_rom\_ena gets asserted, the write process then restarts from address 0 (refer to [Figure 7](#)).

**Figure 7. Asserting the reset\_rom\_address Signal**



## Design Example

You can download design examples for this megafunction from the following locations:

- On the [Documentation: Quartus II Development Software](#) page, in the Using Megafunctions section under I/O
- On the [Documentation: User Guides](#) webpage, with this user guide

The designs are simulated using the ModelSim®-Altera software to generate a waveform display of the device behavior. For more information about the ModelSim-Altera software, refer to the [ModelSim-Altera Software Support](#) page on the Altera website. The support page includes links to such topics as installation, usage, and troubleshooting.

## Frequency Division

This design example uses the ALTPLL\_RECONFIG megafunction to change the clock frequency of an enhanced PLL. This example demonstrates how to reconfigure the c0 counter using the ALTPLL\_RECONFIG megafunction to vary the frequency of this counter by changing the *c* value. [Figure 8](#) shows the formula for changing the *c* value for different PLL output frequencies.

**Figure 8. Frequency Division Formula**

Divide-by value =  $c = (F_{in} * m) / (F_{out} * n)$

Where:

*c* value = High time count = Low time count  
 $F_{in}$  = Input frequency  
 $m$  = *m* modulus value  
 $n$  = *n* modulus value  
 $F_{out}$  = Required output frequency

This example reconfigures the output frequency of c0 from 100 to 50 MHz by changing the divide-by value from 3 to 6.

## Generating the ALTPLL and ALTPLL\_RECONFIG Megafunctions

To generate the ALTPLL and ALTPLL\_RECONFIG megafunctions, follow these steps:

1. Open the `altpll_reconfig_DesignExample_ex1.zip` file and extract `pll_recon_ex1_1.1.qar`.
2. In the Quartus II software, open the `pll_recon_ex1_1.1.qar` file and restore the archive file into your working directory.
3. On the Tools menu, click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
4. Select **Create a new custom megafunction variation**.
5. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears.

6. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings listed in [Table 2](#). Click **Next** to advance from one page to the next.

**Table 2. Configuration Settings for the ALTPLL Megafunction (Part 1 of 2)**

MegaWizard Plug-In Manager Page	Settings	Value
2a	Megafunction	Under the <b>I/O</b> category, select <b>ALTPLL</b>
	Which device family will you be using?	<b>Stratix</b>
	Which type of output file do you want to create?	<b>VHDL</b>
	What name do you want for the output file?	<b>reconfig_pll</b>
	Return to this page for another create operation	Turned on
Parameter Settings (General/Modes)	Currently selected device family	<b>Stratix</b>
	Match project/default	Turned on
	Which device speed grade will you be using?	<b>Any</b>
	What is the frequency of inclk0 input	<b>100 MHz</b>
	Which PLL type will you be using?	<b>Enhanced PLL</b>
	How will the PLL outputs be generated?	Select <b>Use the feedback path inside the PLL</b> . Select <b>In normal mode</b>
	Which output clock will be compensated for?	<b>c0</b>
Parameter Settings (Scan/Inputs/Lock)	Create optional inputs for dynamic reconfiguration	Turned on
	Long chain: All 6 core and 4 external clocks are available	Selected
	Create an 'pllena' input to selectively enable the PLL	Turned off
	Create an 'areset' input to asynchronously reset the PLL	Turned on
	Create an 'pfdena' input to selectively enable the phase/frequency detector	Turned off
	Create 'locked' output	Turned on
	Create output file(s) using 'Advanced' PLL parameters	Turned off
Output Clocks (clk c0)	Use this clock	Turned on
	Enter output clock frequency	<b>100 MHz</b>
	Clock phase shift	<b>0 degrees</b>
	Clock duty cycle (%)	<b>50</b>
	Create a clock enable input	Turned off
EDA	Generate netlist	Turned off

**Table 2. Configuration Settings for the ALTPLL Megafunction (Part 2 of 2)**

MegaWizard Plug-In Manager Page	Settings	Value
Summary	Variation file	Turned on
	PinPlanner ports PPF file	Turned on
	AHDL Include file	Turned on
	VHDL component declaration file	Turned on
	Quartus II symbol file	Turned on
	Instantiation template file	Turned on

7. Click **Finish**. The `reconfig_pll` module is built.
8. Click **OK**. The MegaWizard Plug-In Manager resets to page 2a to allow you to create a new custom megafunction variation.
9. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings listed in [Table 3](#). Click **Next** to advance from one page to the next.

**Table 3. Configuration Settings for the ALTPLL\_RECONFIG Megafunction**

MegaWizard Plug-In Manager Page	Settings	Value
2a	Megafunction	Under the <b>I/O</b> category, select <b>ALTPLL_RECONFIG</b>
	Which device family will you be using?	<b>Stratix</b>
	Which type of output file do you want to create?	<b>VHDL</b>
	What name do you want for the output file?	<b>pll_reconfig</b>
	Return to this page for another create operation	Turned off
Parameter Settings (General)	Currently selected device family	<b>Stratix</b>
	Match project/default	Turned on
	Which scan chain type will you be using	<b>Long chain</b>
Parameter Settings (General 2)	Do you want to specify initial value of the scan chain?	Select <b>Yes, use this file for the content data</b>
	File name	<b>pll_j1_clk0.mif</b>
	Do not use pre initialized RAM - initialize from ROM instead	Turned off
EDA	Generate netlist	Turned off
Summary	Variation file	Turned on
	AHDL Include file	Turned on
	VHDL component declaration file	Turned on
	Quartus II symbol file	Turned on
	Instantiation template file	Turned on

10. Click **Finish**. The `pll_reconfig` module is built.

## Compiling the ALTPLL and ALTPLL\_RECONFIG Megafunctions

To add the ALTPLL megafunction to the ALTPLL\_RECONFIG megafunction, and then compile the design in the Quartus II software, follow these steps:

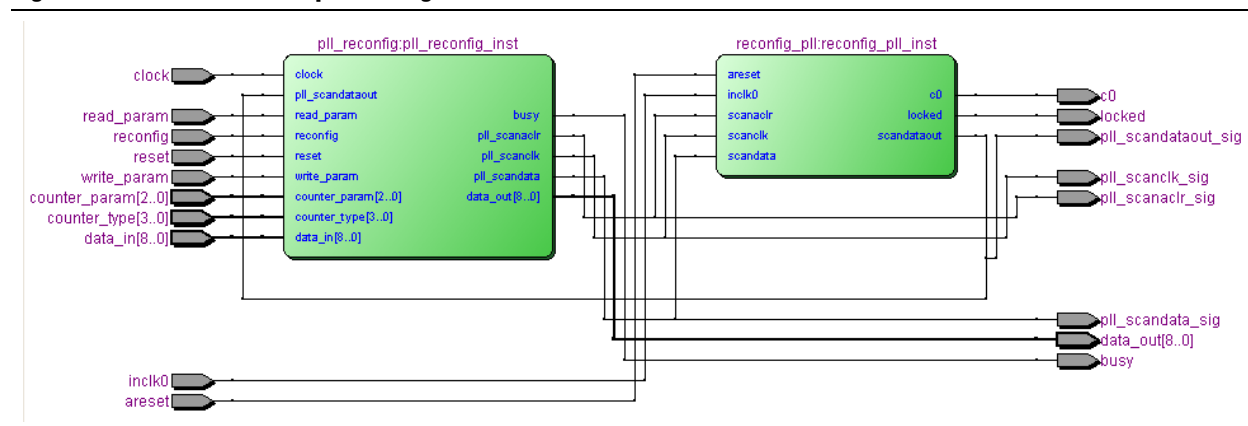
1. On the Project menu, click **Add/Remove File in Project**. The Settings dialog box appears.
2. In the **Category** list, select **Files**.
3. Click **Browse (...)** after **File name** and select **pll\_recon\_ex1.vhd** from the project folder. This file is the top-level module that contains the port-mapping between the pll\_reconfig and reconfig\_pll instances.
4. Click **Add** to add the top-level file to the project.
5. Click **OK**.
6. On the File menu, click **Save Project**.

The top-level file is added to the project.

7. To compile the design, on the Processing menu, click **Start Compilation**.
8. When the **Full Compilation was successful** message box appears, click **OK**.

You have now created and compiled the complete design file, which can be viewed in the RTL Viewer (Figure 9). To display the RTL Viewer, in the Tools menu, select **Netlist Viewers**, and click on **RTL Viewer**.

**Figure 9. RTL Viewer — Complete Design File**



## Simulating the Design Example

To simulate the design example using the ModelSim-Altera software, follow these steps:

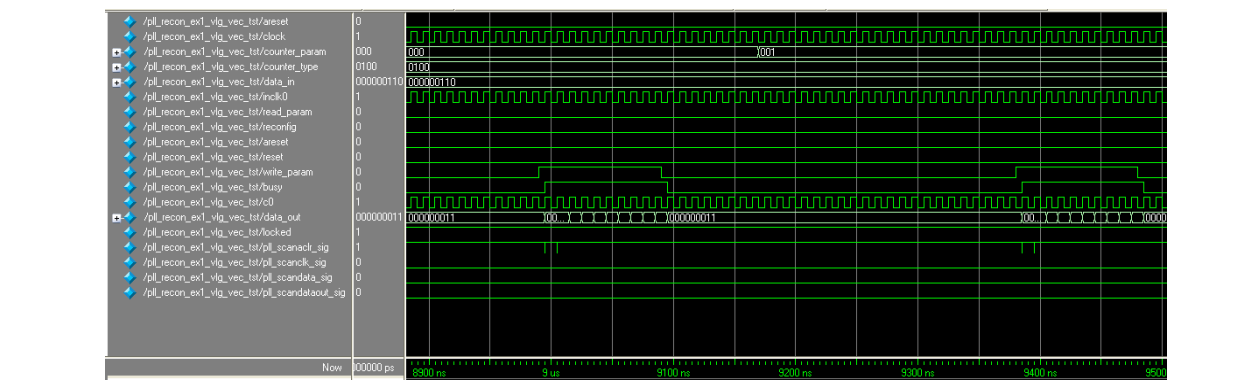
1. Unzip the **altpll\_reconfig\_ex1\_msim.zip** file to any working directory on your PC.
2. Browse to the folder in which you unzipped the files.
3. Open **remote\_update\_ex2.do** in a text editor.
4. In line 1 of the **altpll\_reconfig\_ex1\_msim.do** file, ensure that the directory path of the library files is correct. For example, C:/Modeltech\_ae/altera/verilog/stratix.
5. On the File menu, click **Save**.

6. Launch the ModelSim-Altera software.
7. On the File menu, click **Change Directory**.
8. Select the folder in which you unzipped the files.
9. Click **OK**.
10. On the Tools menu, click **Execute Macro**.
11. Select the **altpll\_reconfig\_ex1\_msim.do** file and click **Open**. This is a script file for ModelSim-Altera software to automate all the necessary settings for the simulation.
12. Verify the results shown in the Wave window.

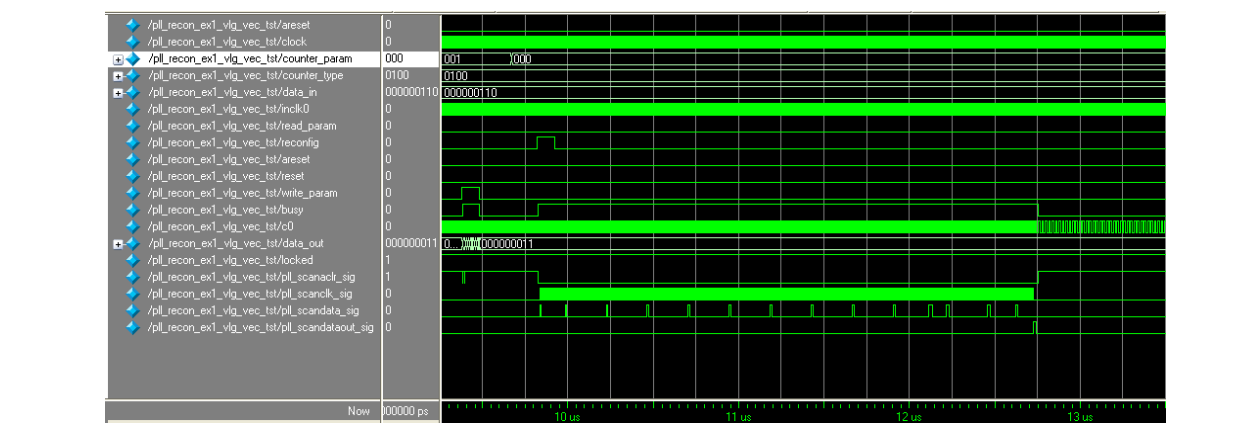
You can rearrange, remove, and add signals, and change the radix by modifying the script **altpll\_reconfig\_ex1\_msim.do**.

Figure 10 and Figure 11 show the expected simulation results in the ModelSim-Altera software. Figure 11 shows the change in c0 frequency starting from 12.75 ms.

**Figure 10. Simulation Results in the ModelSim-Altera Software (8.9 to 9.5 ms)**



**Figure 11. Simulation Results in the ModelSim-Altera Software (9.5 to 13.5 ms)**



## Pulse Width Variation

This design example uses the ALTPLL\_RECONFIG megafunction to modify the pulse width of an enhanced PLL. This example demonstrates how to reconfigure the c1 counter using the ALTPLL\_RECONFIG megafunction to vary the pulse width of this counter by changing the high-count and low-count values. The formula for changing the duty cycle is shown in Figure 12.

**Figure 12. Changing the Duty Cycle Formula**

Duty cycle =  $(Ch/Ct) \% \text{ high time count}$  and  $(Cl/Ct) \% \text{ low time count}$   
with RSELODD = 0

Where:  
Ch = High time count  
Cl = Low time count  
Ct = Total time

When you set RSELODD = 1, you subtract 0.5 cycles from the high time and you add 0.5 cycles to the low time.

For example, if:  
Ch = 2 cycles  
Cl = 1 cycle  
(Note: For odd division factors, the larger number is for the Ch counter; the smaller number is for the Cl counter.)

Setting RSELODD = 1 effectively changes the Ch and Cl to:

High time count = 1.5 cycles  
Low time count = 1.5 cycles

Duty cycle =  $(1.5/3) \% \text{ high time count}$  and  $(1.5/3) \% \text{ low time count}$

In this example, the pulse width is programmed to change from 50% to 25% , and then to 75% of the duty cycle.

## Generating the ALTPLL and ALTPLL\_RECONFIG Megafunctions

To generate the ALTPLL and ALTPLL\_RECONFIG megafunctions, perform the following steps:

1. Open ALTPLL\_RECONFIG\_DesignExample\_ex2.zip and extract pll\_recon\_ex2\_1.1.qar.
2. In the Quartus II software, open pll\_recon\_ex2\_1.1.qar and restore the archive file into your working directory.
3. On the Tools menu, click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
4. Select **Create a new custom megafunction variation**.
5. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears.

6. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings listed in [Table 4](#). Click **Next** to advance from one page to the next.

**Table 4. Configuration Settings for the ALTPLL Megafunction (Part 1 of 2)**

MegaWizard Plug-In Manager Page	Settings	Value
2a	Megafunction	Under the <b>I/O</b> category, select <b>ALTPLL</b>
	Which device family will you be using?	<b>Stratix</b>
	Which type of output file do you want to create?	<b>VHDL</b>
	What name do you want for the output file?	<b>reconfig_pll</b>
	Return to this page for another create operation	Turned on
Parameter Settings (General/Modes)	Currently selected device family	<b>Stratix</b>
	Match project/default	Turned on
	Which device speed grade will you be using?	<b>Any</b>
	What is the frequency of inclk0 input	<b>20 MHz</b>
	Which PLL type will you be using?	<b>Enhanced PLL</b>
	How will the PLL outputs be generated?	Select <b>Use the feedback path inside the PLL</b> . Select <b>In normal mode</b>
	Which output clock will be compensated for?	<b>c1</b>
Parameter Settings (Scan/Inputs/Lock)	Create optional inputs for dynamic reconfiguration	Turned on
	Long chain: All 6 core and 4 external clocks are available	Selected
	Create an 'pllana' input to selectively enable the PLL	Turned off
	Create an 'areset' input to asynchronously reset the PLL	Turned on
	Create an 'pfdena' input to selectively enable the phase/frequency detector	Turned off
	Create 'locked' output	Turned on
	Create output file(s) using 'Advanced' PLL parameters	Turned off
Output Clocks (clk c1)	Use this clock	Turned on
	Enter output clock frequency	<b>15 MHz</b>
	Clock phase shift	<b>0 degrees</b>
	Clock duty cycle (%)	<b>50</b>
	Create a clock enable input	Turned off
EDA	Generate netlist	Turned off



**Table 4. Configuration Settings for the ALTPLL Megafunction (Part 2 of 2)**

MegaWizard Plug-In Manager Page	Settings	Value
Summary	Variation file	Turned on
	PinPlanner ports PPF file	Turned on
	AHDL Include file	Turned on
	VHDL component declaration file	Turned on
	Quartus II symbol file	Turned on
	Instantiation template file	Turned on

- Click **Finish**. The `reconfig_pll` module is built.
- Click **OK**. The MegaWizard Plug-In Manager resets to page 2a to allow you to create a new custom megafunction variation.
- In the MegaWizard Plug-In Manager pages, select or verify the configuration settings listed in [Table 5](#). Click **Next** to advance from one page to the next.

**Table 5. Configuration Settings for the ALTPLL\_RECONFIG Megafunction**

MegaWizard Plug-In Manager Page	Settings	Value
2a	Megafunction	Under the <b>I/O</b> category, select <b>ALTPLL_RECONFIG</b>
	Which device family will you be using?	<b>Stratix</b>
	Which type of output file do you want to create?	<b>VHDL</b>
	What name do you want for the output file?	<b>pll_reconfig</b>
	Return to this page for another create operation	Turned off
Parameter Settings (General)	Currently selected device family	<b>Stratix</b>
	Match project/default	Turned on
	Which scan chain type will you be using	<b>Long chain</b>
Parameter Settings (General)	Do you want to specify initial value of the scan chain?	Select <b>Yes, use this file for the content data</b>
	File name	<b>pll_j1_pll.mif</b>
	Do not use pre initialized RAM - initialize from ROM instead	Turned off
EDA	Generate netlist	Turned off
Summary	Variation file	Turned on
	AHDL Include file	Turned on
	VHDL component declaration file	Turned on
	Quartus II symbol file	Turned on
	Instantiation template file	Turned on

- Click **Finish**. The `pll_reconfig` module is built.

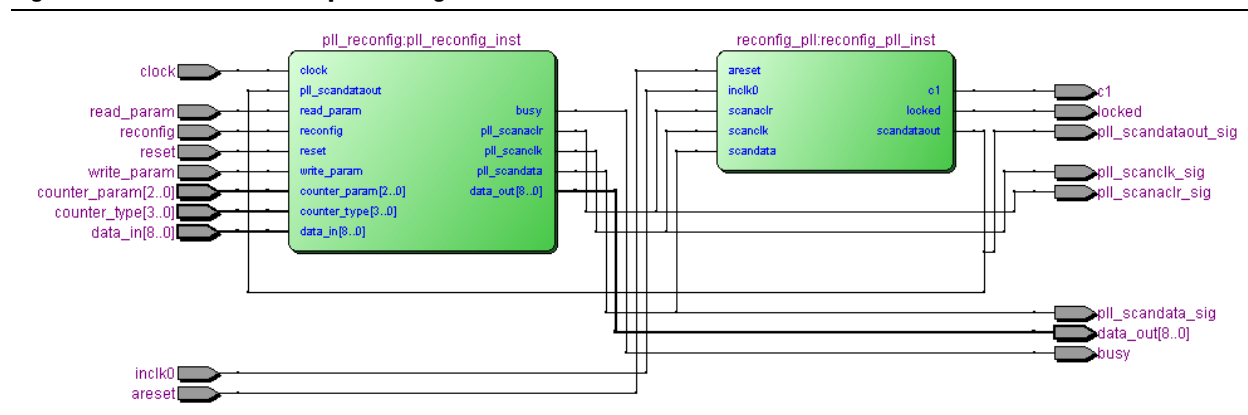
## Compiling the ALTPLL and ALTPLL\_RECONFIG Megafunctions

To add the ALTPLL megafunction to the ALTPLL\_RECONFIG megafunction, and then compile the design in the Quartus II software, follow these steps:

1. On the Project menu, click **Add/Remove Files in Project**. The **Settings** dialog box appears.
2. In the **Category** list, select **Files**.
3. Click **Browse (...)** after **File name** and from the project folder, select **pll\_recon\_ex2.vhd**. This file is the top-level module that contains the port-mapping between the `pll_reconfig` and `reconfig_pll` instances.
4. To add the top-level file to the project, click **Add**.
5. Click **OK**.
6. On the File menu, click **Save Project**.  
The top-level file is added to the project.
7. To compile the design, on the Processing menu, click **Start Compilation**.
8. When the **Full Compilation was successful** message box appears, click **OK**.

You have now created and compiled the complete design file, which can be viewed in the RTL Viewer ([Figure 13](#)). To display the RTL Viewer, in the Tools menu, select **Netlist Viewers**, and click on **RTL Viewer**.

**Figure 13. RTL Viewer — Complete Design File**



## Simulating the Design Example

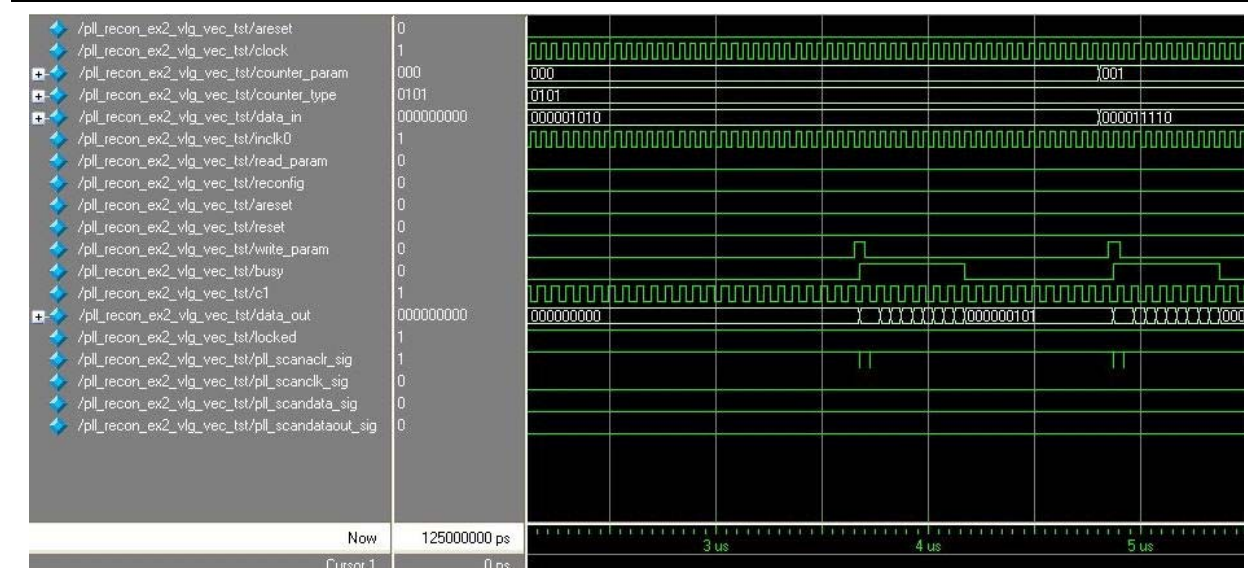
To simulate the design example using the ModelSim-Altera software, follow these steps:

1. Unzip the **altpll\_reconfig\_ex2\_msim.zip** file to any working directory on your PC.
2. Browse to the folder in which you unzipped the files.
3. Open the **remote\_update\_ex2.do** file in a text editor.
4. In line 1 of the **altpll\_reconfig\_ex2\_msim.do** file, make sure the directory path of the library files is correct. For example, C:/Modeltech\_ae/altera/verilog/stratix.
5. On the File menu, click **Save**.
6. Launch the ModelSim-Altera software.
7. On the File menu, click **Change Directory**.
8. Select the folder in which you unzipped the files.
9. Click **OK**.
10. On the Tools menu, click **Execute Macro**.
11. Select the **altpll\_reconfig\_ex2\_msim.do** file and click **Open**. This is a script file for ModelSim-Altera software to automate all of the necessary settings for the simulation.
12. Verify the results shown in the Wave window.

You can rearrange, remove, and add signals. and change the radix by modifying the script **altpll\_reconfig\_ex2\_msim.do**.

Figure 14 through Figure 19 show the expected simulation results in the ModelSim-Altera software. The duty cycle changes from a ratio of 50:50 to 25:75 and finally to 75:25.

**Figure 14. Changing Parameters (2.11 to 7.23 ms)**



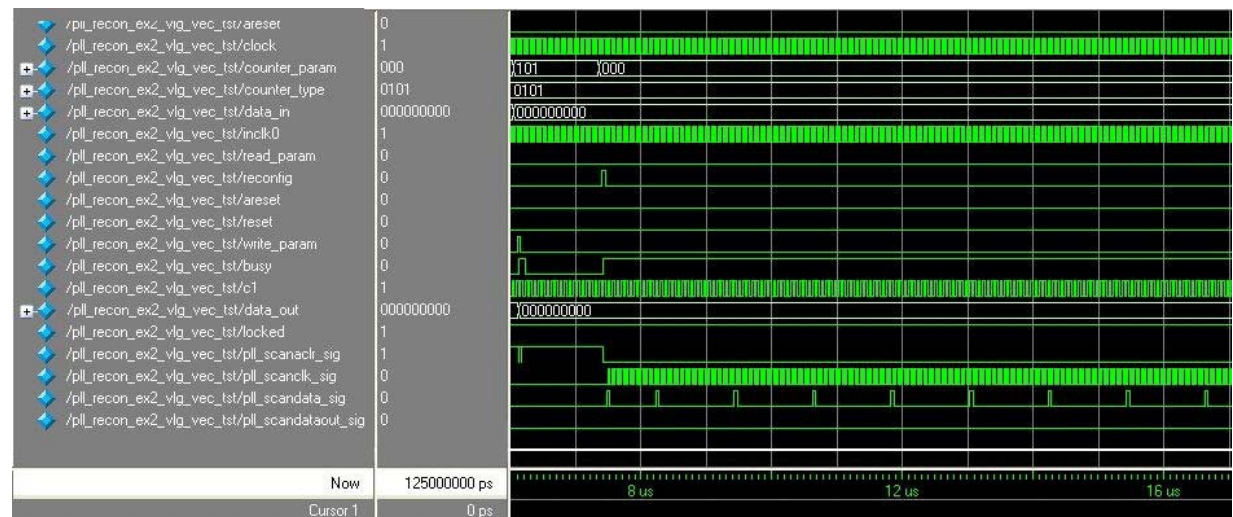
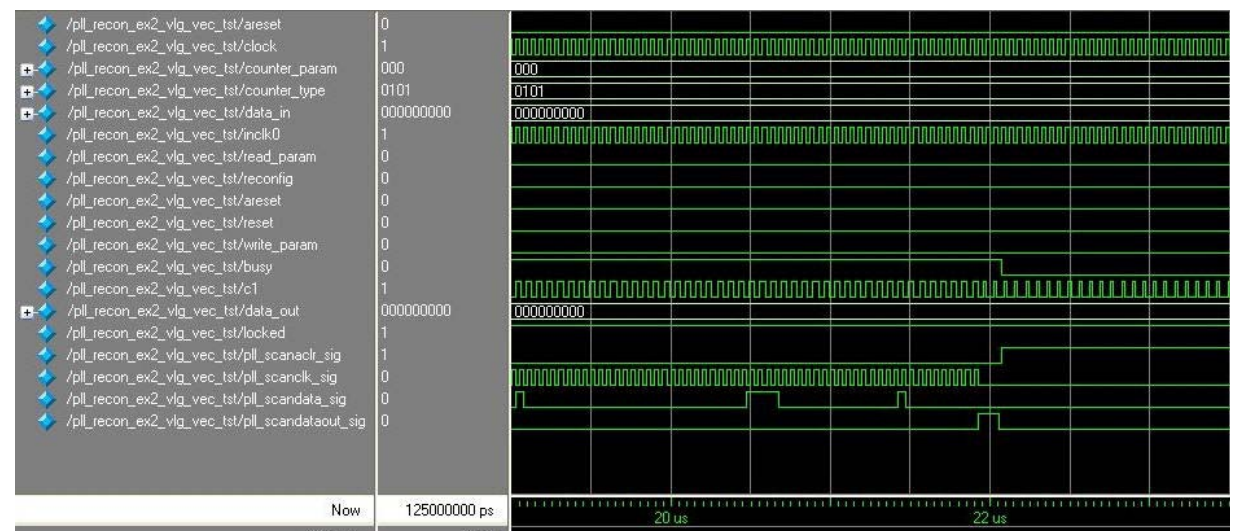
**Figure 15. Reconfiguration (6.32 to 26.8 ms)****Figure 16. Pulse Width Changes From 50:50 Ratio to 25:75 Ratio (20 to 26 ms)**

Figure 17. Changing Parameters (91.79 to 96.91 ms)

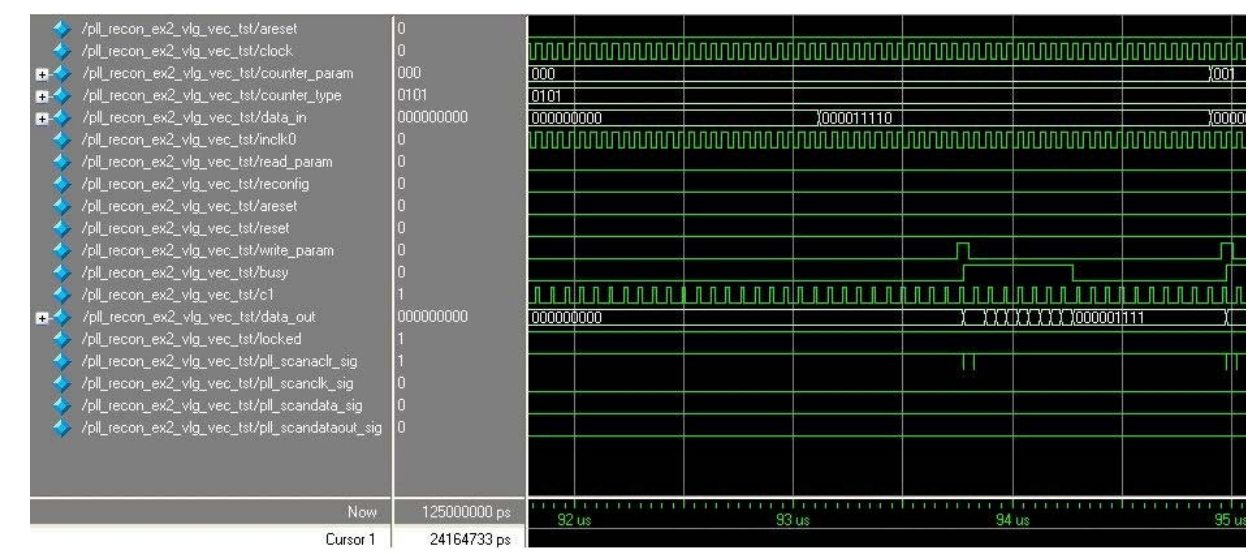


Figure 18. Reconfiguration (96.92 to 117.4 ms)

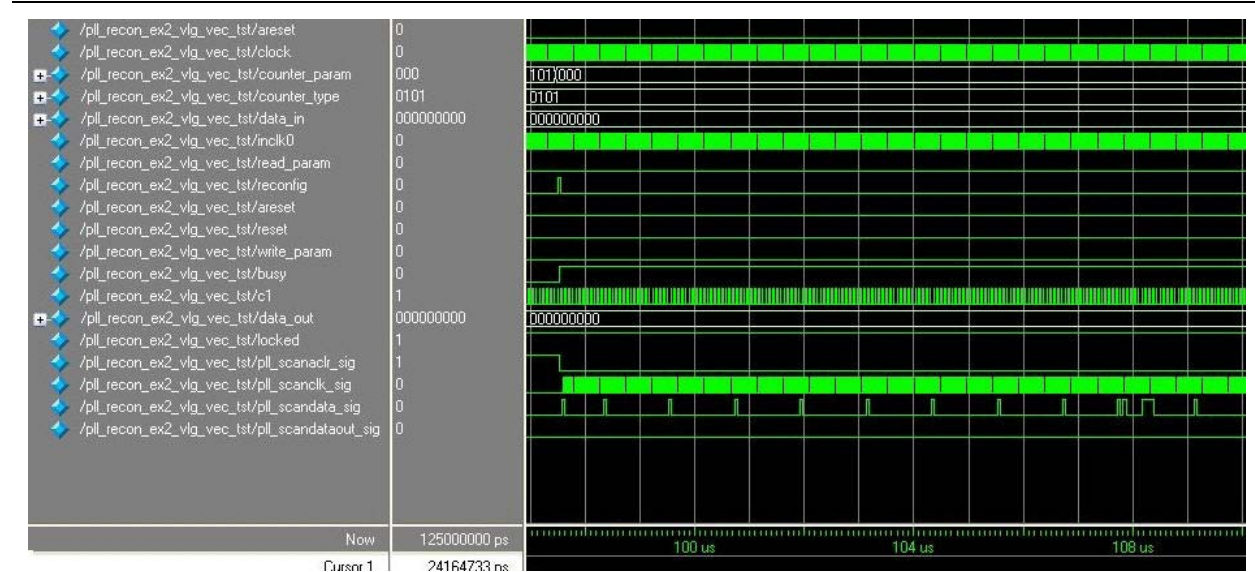
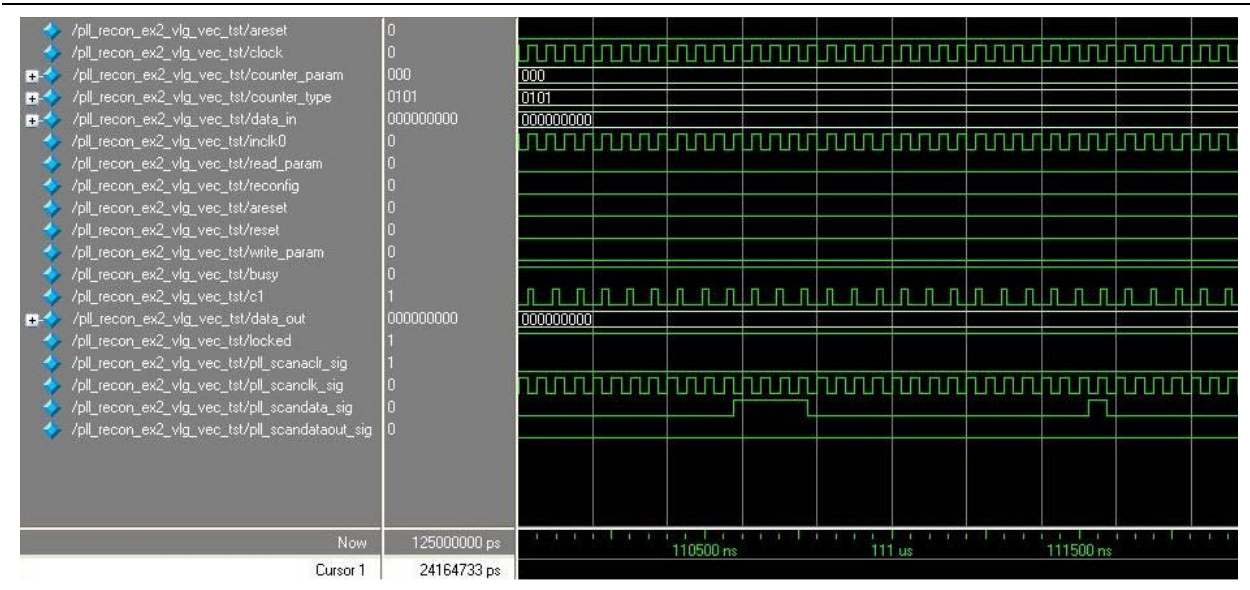




Figure 19. Pulse Width Changes To 75:25 Ratio (110 to 113 ms)



## PLL Reconfiguration with Multiple .mif Files

This design example uses the ALTPLL\_RECONFIG megafunction to reconfigure the output of the c0 counter based on the PLL settings specified in multiple .mif files from external ROMs in Stratix III devices. The .mif files specify PLL settings that reconfigure the output of the c0 counter from 100 to 200 MHz, 300 MHz, 400 MHz, and 500 MHz, then back to 200 MHz.

### Generating the ALTPLL and ALTPLL\_RECONFIG Megafunctions

To generate the ALTPLL and ALTPLL\_RECONFIG megafunctions, follow these steps:

1. Open the **ALTPLL\_RECONFIG\_DesignExample\_ex3.zip** file to any directory on your PC.
2. Open the **ALTPLL\_RECONFIG\_rom.qar** project file.
3. On the Tools menu, click **MegaWizard Plug-In Manager**. Page 1 of the MegaWizard Plug-In Manager appears.
4. Select the **Create a new custom megafunction variation** option.
5. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears.
6. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings listed in [Table 6](#). Click **Next** to advance from one page to the next.

**Table 6. Configuration Settings for the ALTPLL Megafunction (Part 1 of 2)**

MegaWizard Plug-In Manager Page	Settings	Value
2a	Megafunction	Under the I/O category, select <b>ALTPLL</b>
	Which device family will you be using?	<b>Stratix III</b>
	Which type of output file do you want to create?	<b>Verilog</b>
	What name do you want for the output file?	<b>the_pll.v</b>
	Return to this page for another create operation	Turned on
Parameter Settings (General/Modes)	Currently selected device family	<b>Stratix III</b>
	Match project/default	Turned on
	Which device speed grade will you be using?	<b>Any</b>
	What is the frequency of inclk0 input	<b>50 MHz</b>
	Which PLL type will you be using?	<b>Top_Bottom PLL</b>
	How will the PLL outputs be generated?	Select <b>Use the feedback path inside the PLL</b> . Select <b>In normal mode</b>
	Which output clock will be compensated for?	<b>c0</b>

**Table 6. Configuration Settings for the ALTPLL Megafunction (Part 2 of 2)**

MegaWizard Plug-In Manager Page	Settings	Value
Parameter Settings (Scan/Inputs/Lock)	Create an 'pllena' input to selectively enable the PLL	Disabled
	Create an 'areset' input to asynchronously reset the PLL	Turned on
	Create an 'pfdena' input to selectively enable the phase/frequency detector	Turned off
	Create 'locked' output	Turned on
	Enable self reset on loss lock	Turned off
	Create output file(s) using 'Advanced' PLL parameters	Turned off
PLL Reconfiguration	Create optional inputs for dynamic reconfiguration	Turned on
	Initial Configuration File (filename)	<b>the_pll_initial.mif</b> —taking an <i>inclock</i> of 50 MHz and generating <i>c0</i> of 100 MHz Ensure that this option shows the correct path of the .mif file before compiling the design to avoid scan chain mismatch warnings.
	Additional Configuration File (filename)	The files are already generated. They are: <ul style="list-style-type: none"> <li>■ <b>the_pll_200_mhz.mif</b>—taking an <i>inclock</i> of 50 MHz and generating <i>c0</i> of 200 MHz</li> <li>■ <b>the_pll_300_mhz.mif</b>—taking an <i>inclock</i> of 50 MHz and generating <i>c0</i> of 300 MHz</li> <li>■ <b>the_pll_400_mhz.mif</b>—taking an <i>inclock</i> of 50 MHz and generating <i>c0</i> of 400 MHz</li> <li>■ <b>the_pll_500_mhz.mif</b>—taking an <i>inclock</i> of 50 MHz and generating <i>c0</i> of 500 MHz</li> </ul>
	Create optional inputs for dynamic phase reconfiguration	Turned off
Output Clocks (clk c0)	Use this clock	Turned on
	Enter output clock frequency	<b>100 MHz</b>
	Clock phase shift	<b>0 degrees</b>
	Clock duty cycle (%)	<b>50</b>
EDA	Generate netlist	Turned off
Summary	Variation file	Turned on
	PinPlanner ports PPF file	Turned on
	AHDL Include file	Turned on
	VHDL component declaration file	Turned on
	Quartus II symbol file	Turned on
	Instantiation template file	Turned on
	Verilog HDL block box file	Turned on
	Reconfiguration File for altpll_reconfig	Turned on



The ALTPLL megafunction allows you to generate additional configuration files without going through a compilation stage. It allows you to generate as many unique configuration files as you need without the difficulty of multiple compilation flows. All you need to do is to set the intended PLL settings, enter the file name, and click **Generate A Configuration File**. Use this capability with the PLL reconfiguration of multiple **.mif** files via external ROMs in the ALTPLL\_RECONFIG megafunction.

7. Click **Finish**. The **the\_pll.v** module is built.
8. Click **OK**. The MegaWizard Plug-In Manager resets to page 2a so you can create a new custom function variation.
9. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings listed in [Table 7](#). Click **Next** to advance from one page to the next.

**Table 7. Configuration Settings for the ALTPLL\_RECONFIG Megafunction**

MegaWizard Plug-In Manager Page	Settings	Value
2a	Megafunction	Under the <b>I/O</b> category, select <b>ALTPLL_RECONFIG</b>
	Which device family will you be using?	<b>Stratix III</b>
	Which type of output file do you want to create?	<b>Verilog</b>
	What name do you want for the output file?	<b>pll_reconfig_circuit.v</b>
	Return to this page for another create operation	Turned off
Parameter Settings (General)	Currently selected device family	<b>Stratix III</b>
	Match project/default	Turned on
	Which scan chain type will you be using	<b>Top/Bottom</b>
Parameter Settings (General)	Do you want to specify initial value of the scan chain?	Select <b>Yes, use this file for the content data</b>
	File name	<b>the_pll_initial_mif.mif</b> Ensure that this option shows the correct path of the <b>.mif</b> file before compiling the design to avoid scan chain mismatch warnings.
	Do not use pre initialized RAM - initialize from ROM instead	Turned off
	Add ports to write to the scan chain from external ROM during run time	Turned on
EDA	Generate netlist	Turned off
Summary	Variation file	Turned on
	AHDL Include file	Turned on
	VHDL component declaration file	Turned on
	Quartus II symbol file	Turned on
	Instantiation template file	Turned on
	Verilog HDL block-box file	Turned on

10. Click **Finish**. The **pll\_reconfig\_circuit** module is built.

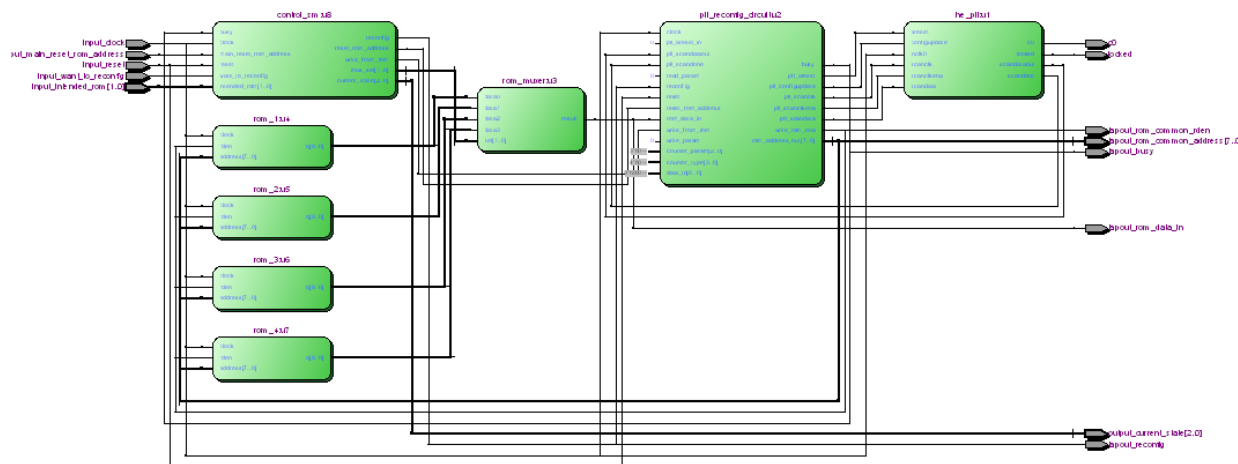
## Compiling the ALTPLL and ALTPLL\_RECONFIG Megafunctions

To add the ALTPLL megafunction to the ALTPLL\_RECONFIG megafunction, and then compile the design in the Quartus II software, follow these steps:

1. On the Project menu, click **Add/Remove Files in Project**. The **Settings** dialog box appears.
2. In the **Category** list, select **Files**.
3. Click **Browse (...)** after **File name** and from the project folder, select **ALTPLL\_RECONFIG\_rom.v**. This file is the top-level module that contains the port-mapping between the `pll_reconfig_circuit` and the `pll` instances.
4. To add the top-level file to the project, click **Add**.
5. Click **OK**.
6. On the File menu, click **Save Project**.  
The top-level file is added to the project.
7. To compile the design, on the Processing menu, click **Start Compilation**.
8. When the **Full Compilation was successful** message box appears, click **OK**.

You have now created and compiled the complete design file, which can be viewed in the RTL Viewer (Figure 20). To display the RTL Viewer, in the Tools menu, select **Netlist Viewers**, and click on **RTL Viewer**.

**Figure 20. Top-Level Design Implementation Using the RTL Viewer**



This design consists of eight modules, which are:

1. **the\_pll:u1**—This represents the Stratix III PLL (Top and Bottom PLL) that is to be reconfigured. The settings are as follows:
  - `inc1k` = 50 MHz
  - `c0` = 100 MHz

2. **pll\_reconfig\_circuit:u2**—This represents the PLL reconfiguration circuitry used by the PLL to reconfigure its settings during user mode. In addition, this circuitry has a scan-chain cache, which contains the intended PLL settings to be reconfigured. It also has additional ports to take advantage of cycling multiple **.mif** files for reconfiguration from external ROMs. This design example demonstrates the capability of these ports. The settings are represented by the **the\_pll\_initial.mif** file. The settings are as follows:
  - `inclk = 50 MHz`
  - `c0 = 100 MHz`
3. **rom\_muxer:u3**—This represents a 4-to-1 multiplexer used to multiplex serial data coming from four ROMs to the `rom_data_in` port of the **pll\_reconfig\_circuit** module. The multiplexer is used because the `rom_data_in` port is 1 bit in size; however, it is controlled by a 2-bit selector, hence its ability to multiplex four signals.
4. **rom\_1:u4**—This represents the external ROM, which contains the intended reconfiguration settings of the PLL. It has a 1-bit output port (`q`) because of the serial nature of writing the intended PLL settings to the scan-chain cache of the **pll\_reconfig\_circuit** module. It has a capacity of 256 words of 1-bit size. The ROM uses 256 words because that is the closest approximate size of the scan-chain file for this type of PLL, which is 234 bits. For this ROM, it is represented by the **the\_pll\_200\_mhz.mif** file, which is 234 bits. The settings are as follows:
  - `inclk = 50 MHz`
  - `c0 = 200 MHz`
5. **rom\_2:u5**—This represents the external ROM, which contains the intended reconfiguration settings of the PLL. It has a 1-bit output port (`q`) because of the serial nature of writing the intended PLL settings to the scan-chain cache of the **pll\_reconfig\_circuit** module. It has a capacity of 256 words of 1-bit size. The ROM uses 256 words because that is the closest approximate size of the scan-chain file for this type of PLL, which is 234 bits. For this ROM, it is represented by the **the\_pll\_300\_mhz.mif** file, which is 234 bits. The settings are as follows:
  - `inclk = 50 MHz`
  - `c0 = 300 MHz`
6. **rom\_3:u6**—This represents the external ROM, which contains the intended reconfiguration settings of the PLL. It has a 1-bit output port (`q`) because of the serial nature of writing the intended PLL settings to the scan-chain cache of the **pll\_reconfig\_circuit** module. It has a capacity of 256 words of 1-bit size. The ROM uses 256 words because that is the closest approximate size of the scan-chain file for this type of PLL, which is 234 bits. For this ROM, it is represented by the **.mif** file **the\_pll\_400\_mhz.mif**, which is 234 bits. The settings are as follows:
  - `inclk = 50 MHz`
  - `c0 = 400 MHz`

7. **rom\_4:u7**—This represents the external ROM, which contains the intended reconfiguration settings of the PLL. It has a 1-bit output port (q) because of the serial nature of writing the intended PLL settings to the scan-chain cache of the **pll\_reconfig\_circuit** module. It has a capacity of 256 words of 1-bit size. The ROM uses 256 words because that is the closest approximate size of the scan-chain file for this type of PLL, which is 234 bits. For this ROM, it is represented by the .mif file **the\_pll\_500\_mhz.mif**, which is 234 bits. The settings are as follows:
  - **inclk** = 50 MHz
  - **c0** = 500 MHz
8. **control\_sm:u8**—This represents the state machine that controls the three main processes involved in the PLL reconfiguration with multiple .mif files via external ROMs. The state machine selects the ROM to be reconfigured, initiates the writing of the ROM content to the scan-chain cache, and initiates the reconfiguration of the PLL using the written content in the scan-chain cache to the PLL. You can modify this simple state machine to suit your design needs.

Figure 21 shows the state diagram for the state machine.

**Figure 21. Control\_sm Module State Diagram**

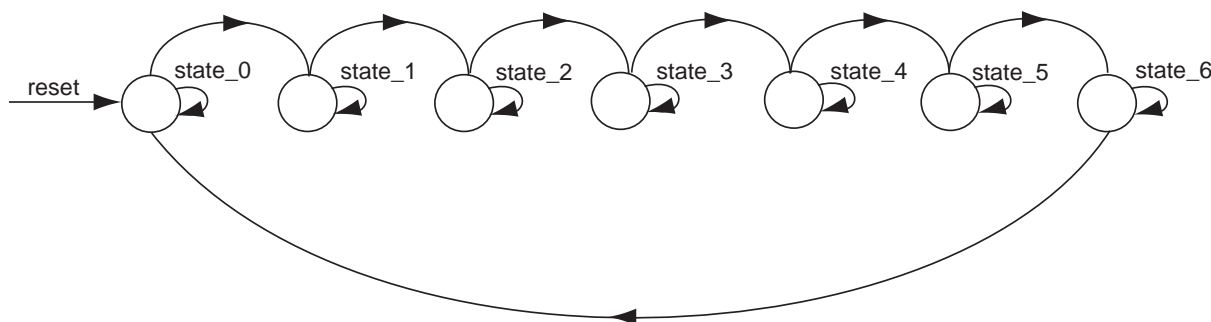


Figure 22 shows the corresponding condition for the state transition.

**Figure 22. Control\_sm Module State Transition Conditions**

	Source State	Destination State	Condition
1	state_0	state_0	(!want_to_reconfig)
2	state_0	state_1	(want_to_reconfig)
3	state_1	state_2	
4	state_2	state_2	(!busy)
5	state_2	state_3	(busy)
6	state_3	state_3	(busy)
7	state_3	state_4	(!busy)
8	state_4	state_5	
9	state_5	state_5	(!busy)
10	state_5	state_6	(busy)
11	state_6	state_0	(!busy)
12	state_6	state_6	(busy)

Figure 23 shows how the whole state machine module (control\_sm) is being implemented in the RTL Viewer.

**Figure 23. control\_sm Design Implementation via the RTL Viewer**

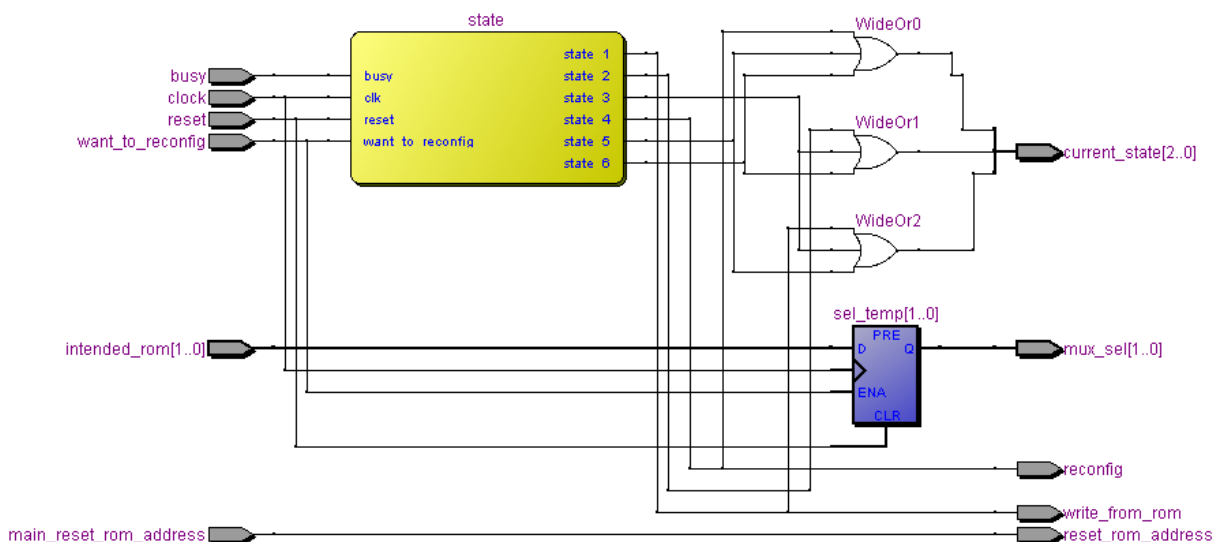


Figure 21 through Figure 23 show how the state machine's control paths and data paths are implemented. The next section describes the state machine behavior in detail.

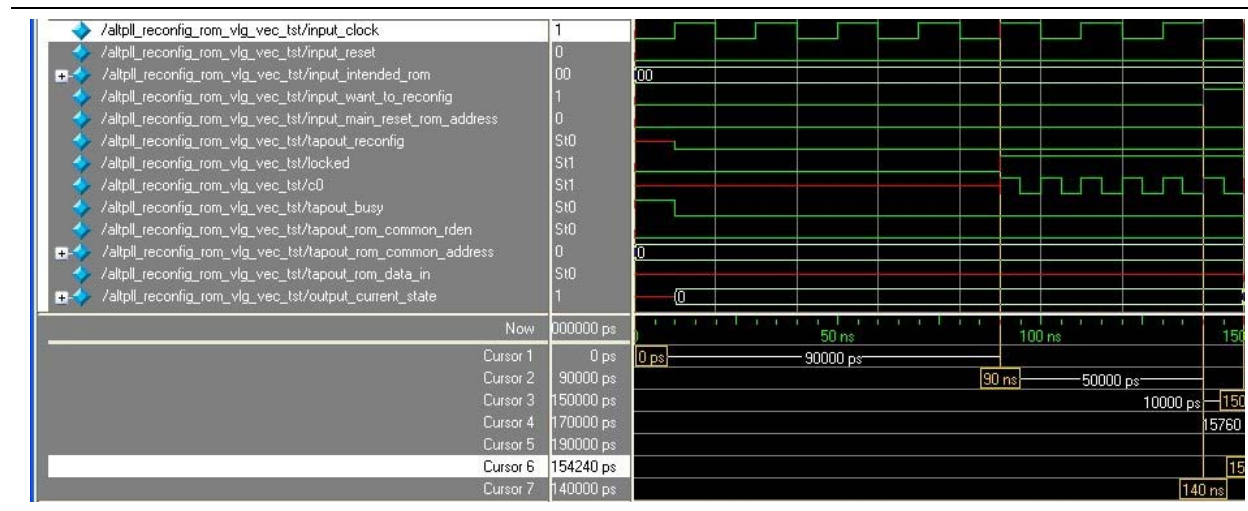
## Simulating the Design Example

To simulate the design example using the ModelSim-Altera software, follow these steps:

1. Unzip the **altpll\_reconfig\_ex3\_msim.zip** file to any directory on your PC.
2. Browse to the folder in which you unzipped the files.
3. Open **remote\_update\_ex2.do** file in a text editor.
4. In line 1 of the **altpll\_reconfig\_ex3\_msim.do**, ensure that the directory path of the library files is correct. For example, C:/Modeltech\_ae/altera/verilog/stratix
5. On the File menu, click **Save**.
6. Launch the ModelSim-Altera software.
7. On the File menu, click **Change Directory**.
8. Select the folder in which you unzipped the files.
9. Click **OK**.
10. On the Tools menu, click **Execute Macro**.
11. Select the **altpll\_reconfig\_ex3\_msim.do** file and click **Open**. This is a script file for the ModelSim-Altera software to automate all of the necessary settings for the simulation.
12. Verify the results shown in the Wave window.

Figure 24 shows the simulation results when writing from ROM 1 to scan cache of the ALTPLL\_RECONFIG megafunction for the duration of 0 to 250 ns.

**Figure 24. Initial Writing from ROM 1 to the Scan Cache of the ALTPLL\_RECONFIG Megafunction (0 to 250 ns)**



The simulation begins when the PLL gets locked (refer to Figure 24); the locked signal is asserted at 90 ns. The PLL output c0 produces a 100 MHz clock. The original settings of the PLL have an input clock of 50 MHz and generates an output clock of 100 MHz.



The output\_current\_state signal is 0, which shows the current state of the state machine that controls the PLL reconfiguration process from the external ROMs.

When the state machine is at 0 (indicated by the output\_current\_state signal), it is waiting for the assertion of the input\_want\_to\_reconfig signal together with the value of the input\_intended\_rom [1:0] signal, which is 00. The state machine remains at this state until the above conditions are satisfied.

At 140 ns, the input\_want\_to\_reconfig signal is asserted for 1 clock cycle and the input\_intended\_rom [1:0] signal is set to 00. The input\_want\_to\_reconfig signal controls the write\_from\_rom signal of the ALTPLL\_RECONFIG instantiation. This begins the process of writing the contents of the intended ROM to the scan cache of the ALTPLL\_RECONFIG megafunction. The input\_intended\_rom [1:0] signal is used to control the selector (sel [1:0] signal) of the multiplexer instantiation, which multiplexes the intended ROM contents (in this case, ROM 1) to the rom\_data\_in signal of the ALTPLL\_RECONFIG instantiation.

At 150 ns, the state machine is at 1 (indicated by the output\_current\_state signal). This signifies that the input\_want\_to\_reconfig signal has been asserted together with the value of the input\_intended\_rom [1:0] signal, which is 00. This causes the write\_from\_rom signal of the ALTPLL\_RECONFIG instantiation to be asserted. This also causes the selector (sel [1:0] signal) of the multiplexer instantiation to multiplex the intended ROM contents (in this case, ROM 1) to the rom\_data\_in signal of the ALTPLL\_RECONFIG instantiation.

At 170 ns, the state machine is at 2 (indicated by the `output_current_state` signal). At this state, the state machine is waiting for the assertion of the `tapout_busy` signal, which signifies the busy signal of the `ALTPLL_RECONFIG` instantiation. The state machine tracks the busy signal because whatever operation the `ALTPLL_RECONFIG` is in (for example, `read_param`, `write_param`, `reconfig`, or `write_from_rom`) when asserted for 1 clock cycle, the busy signal is asserted for a particular duration. This indicates that the particular operation is being processed by the `ALTPLL_RECONFIG` instantiation. In this state, the `tapout_busy` signal has been asserted, signifying that the `ALTPLL_RECONFIG` instantiation has begun processing the `write_from_rom` operation.

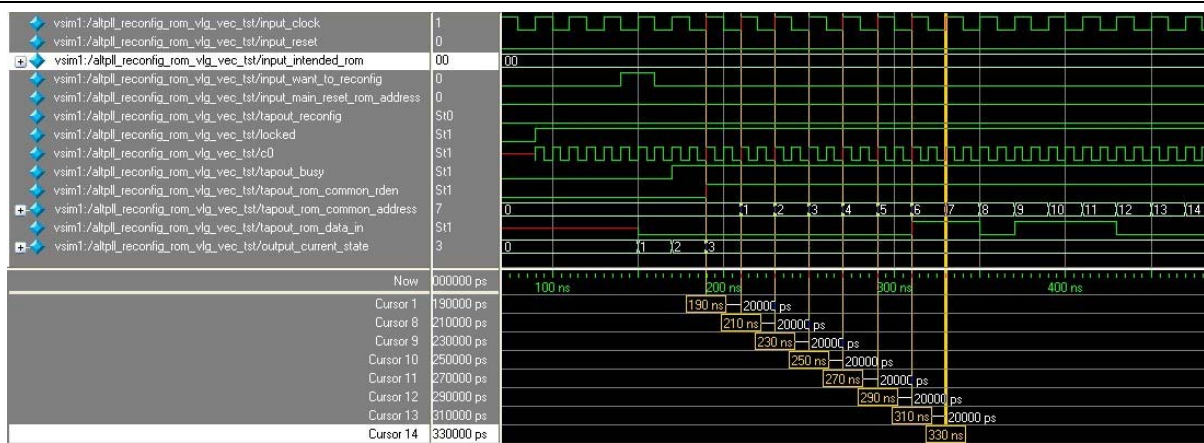
At 190 ns, the state machine is at 3 (indicated by the `output_current_state` signal). This signifies that the `tapout_busy` signal has been asserted. At this point the state machine waits until the `tapout_busy` signal gets deasserted, to signify that the process of writing from the ROM to the scan cache of the `ALTPLL_RECONFIG` instantiation has been completed. Observe that the `tapout_rom_common_rden` signal has been asserted. This is the probed-out signal of the `write_rom_ena` signal, which is part of the `ALTPLL_RECONFIG` instantiation. This signal functions as the enable signal to the ROMs used in this design. Observe that the `tapout_rom_common_address [7:0]` signal begins changing value. This is the probed-out signal of the `rom_address_out [7:0]`, which is part of the `ALTPLL_RECONFIG` instantiation. This signal controls which address of the ROM should be read out to the multiplexer instantiation. When the `tapout_rom_common_rden` signal is asserted together with the value of 0 for the `tapout_rom_common_address [7:0]` signal, it reads out the data from address 0 of the ROM 1 to the `q` port of the ROM, which is connected to the `data_0` signal of the multiplexer. Then the data is multiplexed according to the selector of the multiplexer, and sent out to the `rom_data_in` port of the `ALTPLL_RECONFIG` instantiation. This port is probed out and is observed by the `tapout_rom_data_in` port. Therefore, the data from the intended ROM can be observed in simulation.



The `tapout_rom_common_rden` signal is asserted 1 clock cycle later, after the `tapout_busy` signal is asserted.

Figure 25 shows the simulation results when writing from ROM 1 to the scan cache of the `ALTPLL_RECONFIG` megafunction for the duration of 60 to 580 ns.

**Figure 25. Initial Writing from ROM 1 to the Scan Cache of the ALTPLL\_RECONFIG Megafunction (60 to 580 ns)**



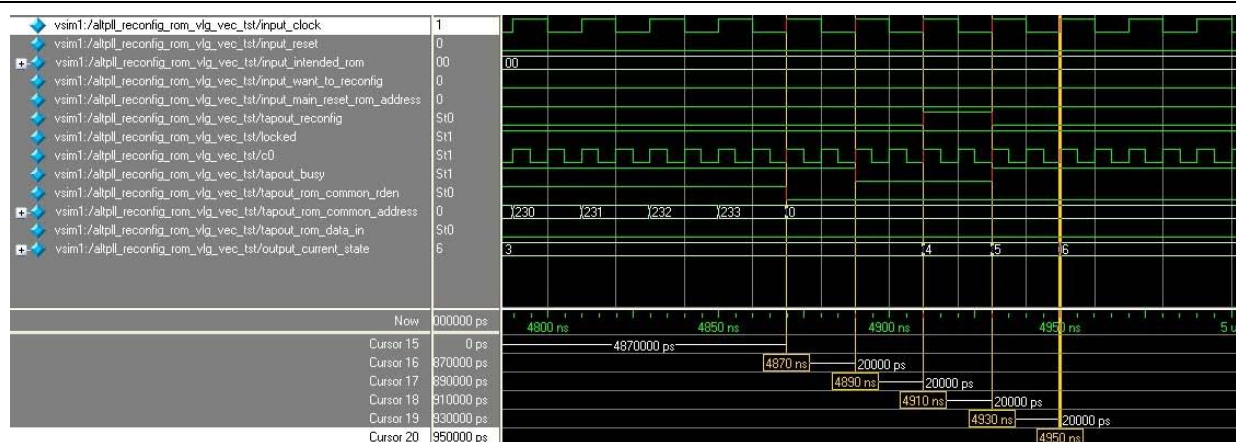


When data from the ROM is written to the scan cache of the ALTPLL\_RECONFIG, it is done serially, meaning a 1-by-1 bit per cycle, from address 0 to address 233. You can observe this via the `tapout_rom_data_in` signal. The valid data read out from the ROM is delayed by 2 clock cycles.

When the simulation is at 270 ns, the `tapout_rom_common_rden` signal is asserted with the `tapout_rom_common_address [7:0]` signal with a value of 4. The data read out has a value of 1 in the `.mif` file in Figure 25. However, this value only appears when the `tapout_rom_common_rden` signal is asserted via the `tapout_rom_common_address [7:0]` signal with a value of 6 at 310 ns, implying a 2-clock-cycle delay.

This writing process continues until it has reached address 233. Figure 26 shows the simulation results of the final process in writing the contents of ROM 1 to the scan cache of the ALTPLL\_RECONFIG megafunction.

**Figure 26. Ending Process of Writing the Contents of ROM 1 to the Scan Cache of the ALTPLL\_RECONFIG Megafunction (4700 to 5040 ns) <sup>(1)</sup>**



**Note to Figure 26:**

- (1) This figure also shows the initialization of the reconfiguration process.

At 4850 ns, the `tapout_rom_common_rden` signal is asserted with the `tapout_rom_common_address [7:0]` signal with a value of 233. This is the last address read out from ROM 1.



The data is available only 2 clock cycles later on the `tapout_rom_data_in` signal. The `tapout_rom_common_rden` signal and the `tapout_busy` signal are still asserted. The `output_current_state` signal still has a value of 3. It will remain in this state until the `tapout_busy` signal is deasserted.

At 4870 ns, the `tapout_rom_common_rden` signal is deasserted with the `tapout_rom_common_address [7:0]` signal with a value of 0. This stops the ROM 1 from reading out data; therefore, the address becomes 0. The `tapout_rom_data_in` signal still generates valid output data and the `tapout_busy` signal is asserted. The `output_current_state` signal still has a value of 3.

At 4890 ns, the `tapout_rom_common_rden` signal remains deasserted with the `tapout_rom_common_address [7:0]` signal with a value of 0. Observe that the `tapout_rom_data_in` signal generates the last valid output data from ROM 1 (from address 233). The `tapout_busy` signal is deasserted and the `output_current_state` signal still has a value of 3.



At 4910 ns, the `tapout_rom_common_rden` signal remains deasserted with the `tapout_rom_common_address [7:0]` signal with a value of 0.



The `tapout_rom_data_in` signal no longer generates valid output data. The `tapout_busy` signal remains deasserted. The `output_current_state` signal changes value from 3 to 4. In this state, the state machine is initiating the reconfiguration process.

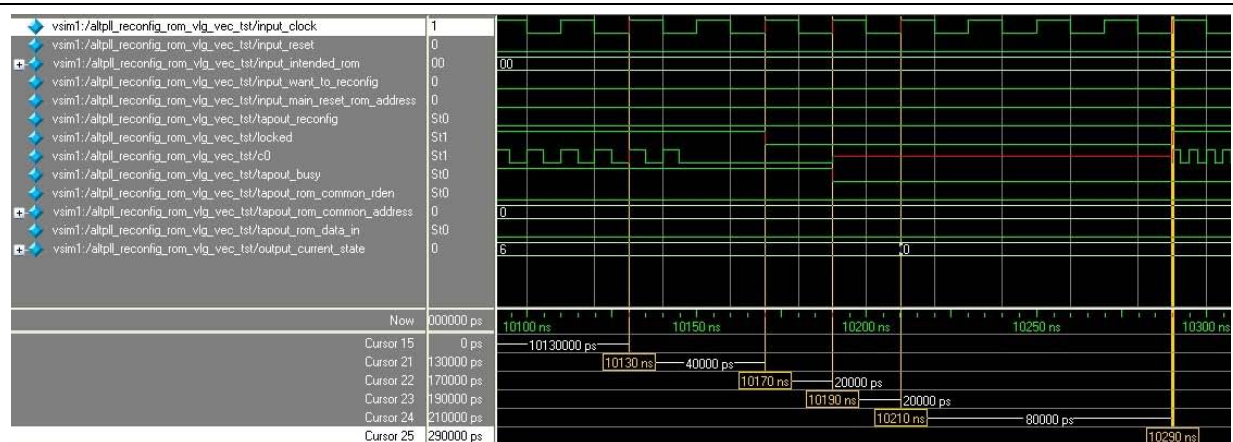
You can observe this state by the assertion of the `tapout_reconfig` signal for 1 clock cycle. This signal controls the `reconfig` signal of the `ALTPLL_RECONFIG` instantiation. When this signal is asserted for 1 clock cycle, the `ALTPLL_RECONFIG` instantiation begins the PLL reconfiguration process by using the settings written from ROM 1 to the scan cache of the `ALTPLL_RECONFIG` instantiation.

At 4930 ns, the `tapout_reconfig` signal is deasserted and the `tapout_busy` signal is asserted. This indicates that the state machine tracks the busy signal. The state machine tracks the busy signal because whatever operation the `ALTPLL_RECONFIG` is in (for example `read_param`, `write_param`, `reconfig`, `write_from_rom`) when asserted for 1 clock cycle, the busy signal is asserted for a particular duration. This indicates that the particular operation is being processed by the `ALTPLL_RECONFIG` instantiation. The `output_current_state` signal changes value from 4 to 5. It remains in this state until the `tapout_busy` signal is asserted.

At 4950 ns, the `tapout_busy` signal remains asserted. The `output_current_state` signal changes value from 5 to 6. It remains in this state until the `tapout_busy` signal is deasserted.

Figure 27 shows the final part of the reconfiguration process.

**Figure 27. PLL Reconfiguration (10,000 to 10,400 ns) <sup>(1)</sup>**



**Note to Figure 27:**

(1) From `c0 = 100 MHz` to `c0 = 200 MHz`.

At 10,130 ns, the `tapout_busy` signal remains asserted. The `output_current_state` signal remains at a value of 6. Therefore, the state machine is still waiting for the `tapout_busy` signal to be deasserted. The `c0` signal is still at 100 MHz and the `locked` signal remains asserted, which means the PLL is still locked to a 100-MHz signal.

At 10,170 ns, the tapout\_busy signal remains asserted. The output\_current\_state signal remains at a value of 6. Therefore, the state machine is still waiting for the tapout\_busy signal to be deasserted. The locked signal is deasserted, which means the PLL has lock loss, and the c0 signal is at a 0 value and not producing a clock pulse.

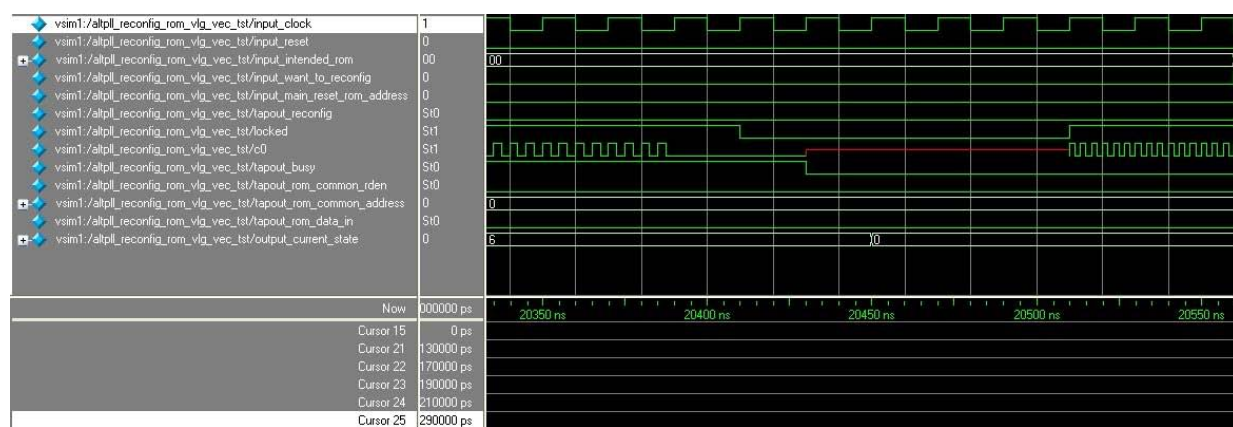
At 10,190 ns, the tapout\_busy signal is deasserted. This means the PLL reconfiguration process is complete. The output\_current\_state signal remains at a value of 6. The locked signal remains deasserted. The c0 signal is an unknown value and still not producing a clock pulse.

At 10,210 ns, the tapout\_busy signal remains deasserted. The output\_current\_state signal changes to a value of 0. This is the original state, in which the state machine waits for the next reconfiguration from an external ROM. The locked signal remains deasserted. The c0 signal is an unknown value and still does not produce a clock pulse.

At 10,290 ns, the tapout\_busy signal remains deasserted. The output\_current\_state signal remains at a value of 0. The locked signal is asserted. The c0 signal now produces a 200-MHz clock signal, which is the intended setting from ROM 1.

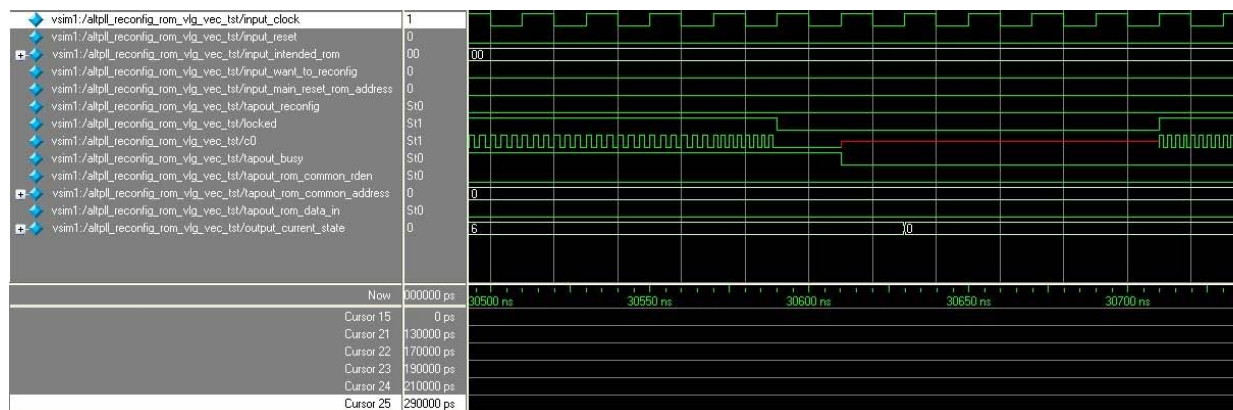
Figure 28 through Figure 30 show the PLL reconfiguration process from the remaining ROMs (ROM 2, ROM 3, and ROM 4, respectively).

**Figure 28. PLL Reconfiguration from ROM 2 (20,330 to 20,600 ns) (1)**

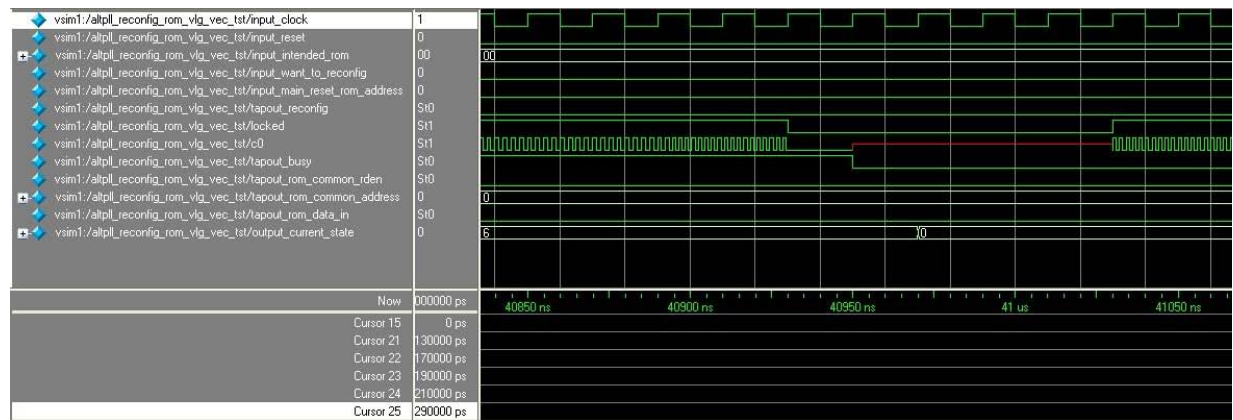


**Note to Figure 28:**

(1) From c0 = 200 MHz to c0 = 300 MHz.

**Figure 29. PLL Reconfiguration from ROM 3 (30,480 to 30,750 ns) (1)****Note to Figure 29:**

(1) From c0 = 300 MHz to c0 = 400 MHz.

**Figure 30. PLL Reconfiguration from ROM 4 (40,800 to 41,090 ns) (1)****Note to Figure 30:**

(1) From c0 = 400 MHz to c0 = 500 MHz.

The next part of the simulation demonstrates the PLL reconfiguration from ROM 1 again, but highlights the ROM address resetting capabilities during writing from an external ROM to the scan cache of the ALTPLL\_RECONFIG instantiation. The c0 signal is 500 MHz and is reconfigured to 100 MHz. Figure 31 shows the process of initiating writing the contents of ROM 1 to the ALTPLL\_RECONFIG instantiation.

**Figure 31. Initial Writing from ROM 1 to the Scan Cache of the ALTPLL\_RECONFIG Megafunction (41,070 to 41,330 ns)**

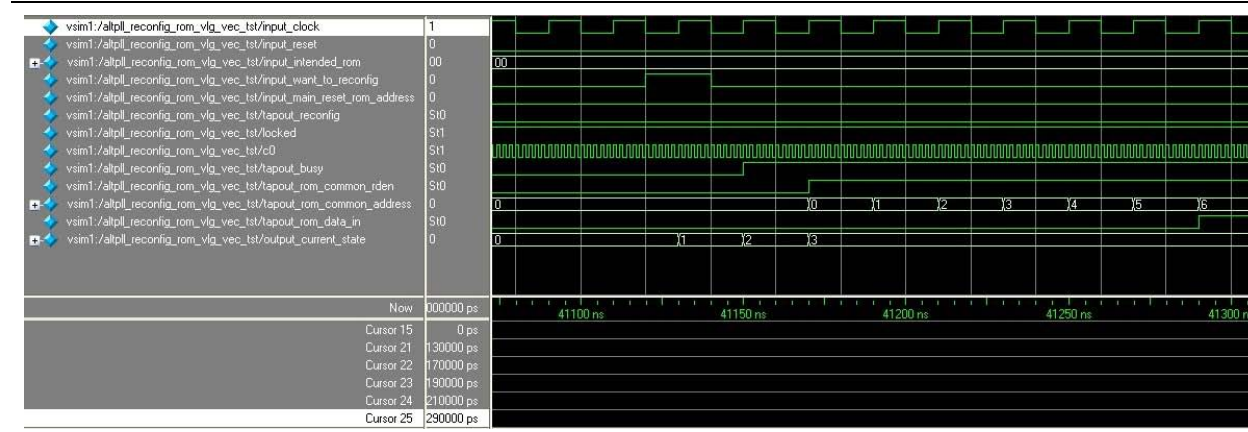
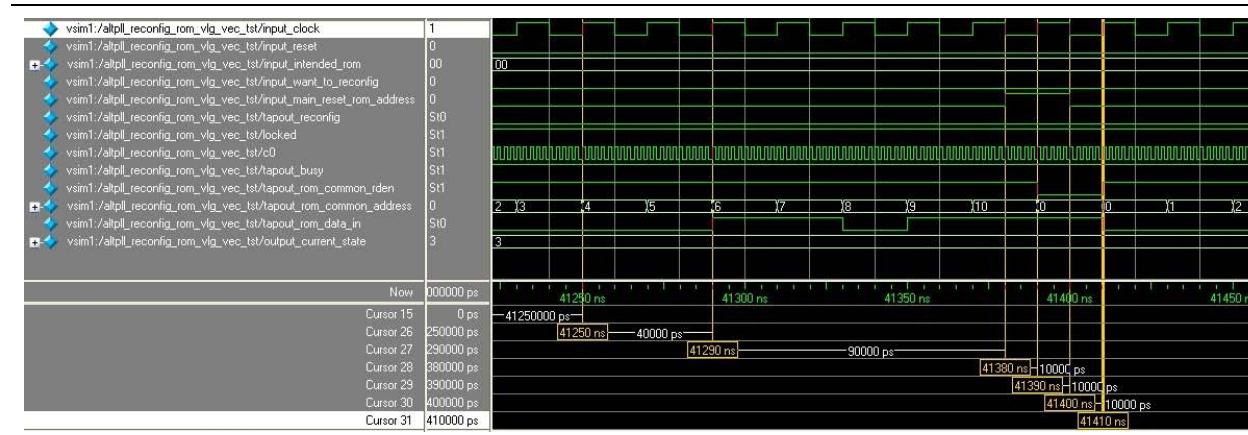


Figure 32 shows how the reset\_rom\_address capability is used.

**Figure 32. Resetting the Address when Writing from ROM 1 to the Scan Cache of the ALTPLL\_RECONFIG Megafunction (41,200 to 41,450 ns)**



At 41,250 ns, the tapout\_rom\_common\_rden signal is asserted with the tapout\_rom\_common\_address [7:0] signal with a value of 4. Observe that the tapout\_busy signal is asserted. The output\_current\_state signal is 3. The normal writing process from ROM to scan cache of the ALTPLL\_RECONFIG instantiation continues.

At 41,290 ns, the tapout\_rom\_common\_rden signal is asserted with the tapout\_rom\_common\_address [7:0] signal with a value of 6. The tapout\_rom\_data\_in signal contains the valid data from address 4 because of the 2-clock-cycle delay. The tapout\_busy signal is asserted. The output\_current\_state signal is 3.

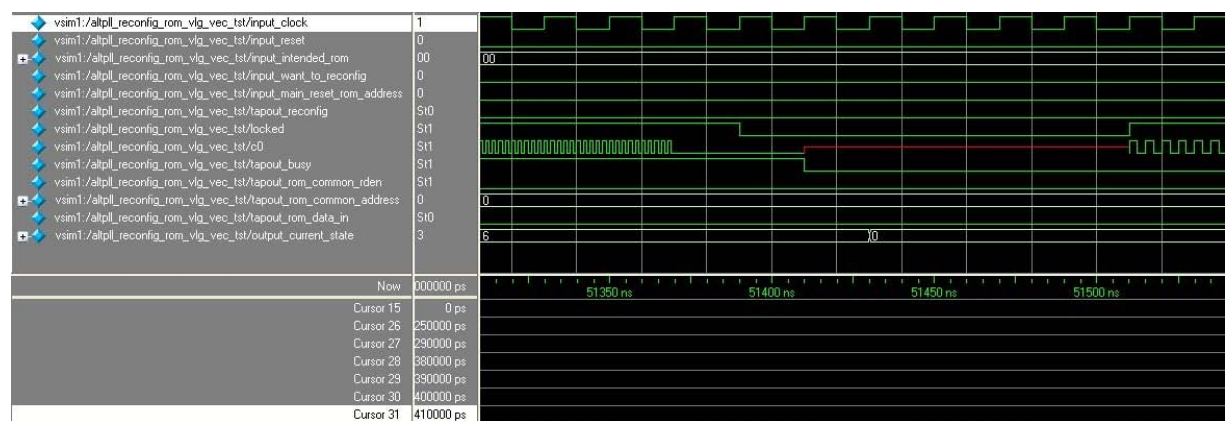
At 41,380 ns, the input\_main\_reset\_rom\_address is asserted for 1 clock cycle. This signal controls the reset\_rom\_address of the ALTPLL\_RECONFIG instantiation. It resets the address counter in the ALTPLL\_RECONFIG instantiation to 0.

At 41,390 ns, the `tapout_rom_common_rden` signal is deasserted for 1 clock cycle. The `tapout_rom_common_address [7:0]` signal is changed to a value of 0. The `tapout_rom_data_in` signal still generates the previous 2-clock-cycle output data (address 9), which is of value 1. The `tapout_busy` signal is still asserted. The `output_current_state` signal is 3.

At 41,410 ns, the `tapout_rom_common_rden` signal is reasserted. The `tapout_rom_common_address [7:0]` signal is changed to a value of 0. This restarts the writing of the contents of ROM 1 to the ALTPLL\_RECONFIG scan cache from address 0. The `tapout_busy` signal is still asserted. The `output_current_state` signal is 3.

The PLL reconfiguration process continues as normal until the `c0` signal is 200 MHz, as shown in Figure 33.

**Figure 33. PLL Reconfiguration From ROM1 (40,800 to 41,090 ns)** (1)



**Note to Figure 33:**

(1) From `c0` = 500 MHz to `c0` = 200 MHz.

You can modify the design to suit your requirements when attempting to reconfigure the PLL from multiple `.mif` files via external ROMs.

## Specifications

This section describes the prototypes, component declarations, ports, and parameters of the ALTPLL\_RECONFIG megafunction. These ports and parameters are available to customize the ALTPLL\_RECONFIG megafunction according to your application.

### Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (`.v`) `altera_mf.v` in the `<Quartus II installation directory>\eda\synthesis` directory.

```
modulealtpll_reconfig
#( parameterintended_device_family = "unused",
  parameterinit_from_rom = "NO",
  parameterpll_type = "UNUSED",
  parameterscan_chain = "UNUSED",
  parameterscan_init_file = "UNUSED",
```



```

        parameter use_scanclk_sync_register = "NO",
        parameter lpm_type = "altpll_reconfig",
        parameter lpm_hint = "unused")
    (
        outputwirebusy,
        inputwireclock,
        inputwire[2:0]counter_param,
        inputwire[3:0]counter_type,
        inputwire[8:0]data_in,
        outputwire[8:0]data_out,
        outputwirepll_areset,
        inputwirepll_areset_in,
        outputwirepll_configupdate,
        outputwirepll_scanaclr,
        outputwirepll_scanclk,
        outputwirepll_scanclkena,
        outputwirepll_scandata,
        inputwirepll_scandataout,
        inputwirepll_scandone,
        outputwirepll_scanread,
        outputwirepll_scanwrite,
        inputwireread_param,
        inputwirereconfig,
        inputwirereset,
        inputwirereset_rom_address,
        outputwire[7:0]rom_address_out,
        inputwirerom_data_in,
        inputwirewrite_from_rom,
        inputwirewrite_param,
        outputwirewrite_rom_ena)/* synthesis syn_black_box=1 */;

endmodule //altpll_reconfig

```

## VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (.vhd) **altera\_mf\_components.vhd** in the *<Quartus II installation directory>\libraries\vhdl\altera\_mf* directory.

```

component altpll_reconfig
    generic (
        intended_device_family:string := "unused";
        init_from_rom:string := "NO";
        pll_type:string := "UNUSED";

```

```
scan_chain:string := "UNUSED";
scan_init_file:string := "UNUSED";
use_scanclk_sync_register:string := "NO";
lpm_hint:string := "UNUSED";
lpm_type:string := "altpll_reconfig"
);
port(
    busy: out std_logic;
    clock: in std_logic;
    counter_param:in std_logic_vector(2 downto 0) := (others =>
'0');
    counter_type:in std_logic_vector(3 downto 0) := (others =>
'0');
    data_in:in std_logic_vector(8 downto 0) := (others => '0');
    data_out:out std_logic_vector(8 downto 0);
    pll_areset:out std_logic;
    pll_areset_in:in std_logic := '0';
    pll_configupdate:out std_logic;
    pll_scanaclr:out std_logic;
    pll_scanclk:out std_logic;
    pll_scanclkena:out std_logic;
    pll_scandata:out std_logic;
    pll_scandataout:in std_logic := '0';
    pll_scandone:in std_logic := '0';
    pll_scanread:out std_logic;
    pll_scanwrite:out std_logic;
    read_param:in std_logic := '0';
    reconfig:in std_logic := '0';
    reset: in std_logic;
    reset_rom_address:in std_logic := '0';
    rom_address_out:out std_logic_vector(7 downto 0);
    rom_data_in:in std_logic := '0';
    write_from_rom:in std_logic := '0';
    write_param:in std_logic := '0';
    write_rom_ena:out std_logic
);
end component;
```

## Ports and Parameters

This section describes the ports and parameters of the ALTPLL\_RECONFIG megafunction.

Table 8 lists the ALTPLL\_RECONFIG megafunction input ports.

**Table 8. ALTPLL\_RECONFIG Megafunction Input Ports (Part 1 of 3)**

Port Name	Required?	Description
clock	Yes	<p>Clock input for loading individual parameters. This signal also clocks the PLL during reconfiguration.</p> <p>The clock input port must be connected to a valid clock.</p> <p>Refer to the <i>DC and Switching Characteristics</i> chapter of the respective device handbooks for the clock <math>f_{MAX}</math>.</p>
reset	Yes	<p>Asynchronous reset input to the megafunction.</p> <p>Altera recommends that you reset this megafunction before first use to guarantee that it is in a valid state. However, it does power up in the reset state. This port must be connected.</p>
data_in[]	No	<p>Data input that provides parameter value when writing parameters.</p> <p>A 9-bit input port that provides the data to be written to the scan cache during a write operation. The bit width of the counter parameter to be written determines the number of bits of data_in[] that are read into the cache. For example, the low bit count of the C0 counter is 8-bit wide, so data_in[7..0] is read to the correct cache location. The bypass mode for the C0 counter is 1-bit wide, so data_in[0] is read for the value of this parameter. If omitted, the default value is 0.</p>
counter_type[]	No	<p>Specifies the counter type.</p> <p>An input port in the form of a 4-bit bus that selects which counter type should be selected for the corresponding operation (read, write, or reconfig). The following table specifies the mapping between the counter_type value and the physical counter to be set. For details, refer to the following tables:</p> <ul style="list-style-type: none"> <li>■ <a href="#">Table 11 on page 45</a> counter_type[3..0] settings for Stratix III, Stratix IV, and Cyclone III devices.</li> <li>■ <a href="#">Table 13 on page 47</a> counter_type[3..0] settings for Stratix II, Stratix II GX, Arria GX and HardCopy II devices.</li> <li>■ <a href="#">Table 14 on page 49</a> counter_type[3..0] settings for Stratix and Stratix GX devices.</li> </ul>
counter_param[]	No	<p>Specifies the parameter for the value specified in the counter_type port.</p> <p>An input port in the form of a 3-bit bus that selects which parameter for the given counter type should be updated. The mapping to each parameter type and the corresponding parameter bit-width are defined in the following tables:</p> <ul style="list-style-type: none"> <li>■ <a href="#">Table 12 on page 46</a> counter_param[2..0] settings for Stratix III, Stratix IV, and Cyclone III devices.</li> <li>■ <a href="#">Table 13 on page 47</a> counter_param[2..0] settings for Stratix II, Stratix II GX, Arria GX and HardCopy II devices.</li> <li>■ <a href="#">Table 14 on page 49</a> counter_param[2..0] settings for Stratix and Stratix GX devices.</li> </ul>



**Table 8. ALTPLL\_RECONFIG Megafunction Input Ports (Part 2 of 3)**

Port Name	Required?	Description
read_param	No	<p>Reads the parameter specified with the <code>counter_type</code> and <code>counter_param</code> ports from cache and fed to the <code>data_out[]</code> port.</p> <p>When asserted, the <code>read_param</code> signal indicates that the scan cache should be read and fed to <code>data_out[]</code>. The bit location of the scan cache and the number of bits read and sent to <code>data_out[]</code> depend on the <code>counter_type</code> and <code>counter_param</code> values. The <code>read_param</code> signal is sampled at the rising clock edge. If it is asserted, the parameter value is read from the cache. Assert the <code>read_param</code> signal for 1 clock cycle only to prevent the parameter from being read twice.</p> <p>The <code>busy</code> signal is asserted on the rising clock edge following the assertion of <code>read_param</code>. While the parameter is being read, the <code>busy</code> signal remains asserted. After the <code>busy</code> signal is deasserted, the value on <code>data_out[]</code> is valid and the next parameter can be loaded. While the <code>busy</code> signal is asserted, the value on <code>data_out[]</code> is not valid.</p> <p>When the <code>read_param</code> signal is asserted, the <code>busy</code> signal is only asserted on the following rising edge of the clock and not on the same clock cycle as <code>read_param</code>.</p>
write_param	No	<p>Writes the parameter specified with the <code>counter_type</code> and <code>counter_param</code> ports to the cache with the value specified on the <code>data_in[]</code> port.</p> <p>When asserted, the <code>write_param</code> signal indicates that the value on <code>data_in[]</code> should be written to the parameter specified by <code>counter_type[]</code> and <code>counter_param[]</code>. The number of bits read from the <code>data_in[]</code> port depends on the parameter. The <code>write_param</code> signal is sampled at the rising clock edge. If it is asserted, the parameter value is written to the cache. Assert the <code>write_param</code> signal for 1 clock cycle only to prevent the parameter from being written twice.</p> <p>The <code>busy</code> signal is asserted on the rising clock edge following the assertion of <code>write_param</code>. While the parameter is being written, the <code>busy</code> signal remains asserted and input to <code>data_in[]</code> is ignored. After the <code>busy</code> signal is deasserted, the next parameter can be written.</p> <p>When the <code>write_param</code> signal is asserted, the <code>busy</code> signal is only asserted on the following rising edge of the clock and not on the same clock cycle as <code>write_param</code>.</p>
reconfig	Yes	<p>Specifies that the phase-locked loop (PLL) should be reconfigured with the PLL settings specified in the current cache.</p> <p>When asserted, the <code>reconfig</code> signal indicates that the PLL should be reconfigured with the values in the cache. The <code>reconfig</code> signal is sampled at the rising clock edge. If it is asserted, the cached settings are loaded in the PLL. Assert the <code>reconfig</code> signal for 1 clock cycle only to prevent reloading the PLL configuration. The <code>busy</code> signal is asserted on the rising clock edge following the assertion of <code>reconfig</code>. While the PLL is being loaded, the <code>busy</code> signal remains asserted. After the <code>busy</code> signal is deasserted, the parameter values can be modified again.</p> <p>During and after reconfiguration, the scan chain data cache remains unchanged. This allows you to easily create a new set of reconfiguration settings that differs from the previous one in only one parameter.</p> <p>If <code>write_param</code> has not been asserted since the previous assertion of <code>reconfig</code>, in Stratix II devices, the <code>pll_scanwrite</code> signal is pulsed during reconfiguration. This feature supports burst-phase stepping. However, in Stratix III, Stratix IV, Cyclone III, Cyclone IV, and Arria II GX devices, the entire scan chain is shifted in to the PLL again.</p> <p>When the <code>reconfig</code> signal is asserted, the <code>busy</code> signal is only asserted on the following rising edge of the clock and not on the same clock cycle as <code>reconfig</code>.</p>

**Table 8. ALTPLL\_RECONFIG Megafunction Input Ports (Part 3 of 3)**

Port Name	Required?	Description
<code>pll_areset_in</code>	No	<p>Input signal indicating that the PLL should be reset.</p> <p>When asserted, the <code>pll_areset_in</code> signal indicates the PLL megafunction should be reset. This port defaults to 0 if left unconnected. When the ALTPLL_RECONFIG megafunction is used in a design, you cannot reset the PLL in any other way; you must use this megafunction port to manually reset the PLL.</p>
<code>pll_scandone</code>	No	<p>Input port for the ALTPLL_RECONFIG megafunction that signals update done from the PLL module. This port is driven by the PLL's <code>scandone</code> output signal and determines when the PLL is reconfigured.</p> <p>For Stratix III and Stratix IV devices, <code>pll_scandone</code> is asserted on most operations, and deasserted as soon as it detects the assertion of <code>configupdate</code>. Upon the completion of all PLL reconfiguration updates, <code>scandone</code> is reasserted.</p> <p>This is different than the <code>scandone</code> signal in older device families, such as Stratix devices. In Stratix devices, the <code>scandone</code> signal is deasserted most of the time. The <code>scandone</code> signal is asserted for 1 clock cycle to mark the completion of PLL reconfiguration.</p>
<code>pll_scandataout</code>	Yes	<p>Input port driven by the <code>scandataout</code> signal from the ALTPLL megafunction. It can be used to read the current configuration of the ALTPLL megafunction.</p> <p>This input port holds the ALTPLL megafunction scan data output from the dynamically reconfigurable bits. The <code>pll_scandataout</code> port must be connected to the <code>scandataout</code> port of the PLL. The activity on this port can only be observed when the <code>reconfig</code> signal is asserted.</p>
<code>rom_data_in</code>	No	<p>Serial data input to the PLL from the ROM.</p> <p>This 1-bit port allows the external ROM to be serially written into the scan cache of the ALTPLL_RECONFIG megafunction. The value on this port is valid 2 clock cycles after the <code>write_rom_ena</code> signal is asserted and remains valid until 2 clock cycles after the <code>write_rom_ena</code> signal is deasserted. This port is available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices only.</p>
<code>write_from_rom</code>	No	<p>Specifies that the scan chain must be written from the external ROM.</p> <p>When asserted, this signal tells the ALTPLL megafunction to write the scan cache from the external ROM. At the next rising clock edge, the ALTPLL_RECONFIG megafunction asserts the <code>write_rom_ena</code> signal and begins placing valid ROM addresses on the <code>rom_address_out</code> port. Assert the <code>write_from_rom</code> signal for 1 clock cycle only. This port is available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices only.</p>
<code>reset_rom_address</code>	No	<p>Resets the ROM address.</p> <p>When asserted, this signal tells the ALTPLL megafunction to restart the read from ROM at address 0. Assert the <code>reset_rom_address</code> signal for 1 clock cycle only. This port is available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices only.</p>

Table 9 lists the ALTPLL\_RECONFIG megafunction output ports.

**Table 9. ALTPLL\_RECONFIG Megafunction Output Ports (Part 1 of 2)**

Port Name	Required?	Description
data_out[]	No	Data read from the cache when read_param is asserted. 9-bit output bus that provides the parameter data to the user. When the read_param signal is asserted, the values on counter_type[] and counter_param[] determine the parameter value that is loaded from cache and driven on the data_out[] bus. When the megafunction deasserts the busy signal, the appropriate bits of the bus (for example, [0] or [3..0]) hold a valid value.
busy	No	Indicates when the PLL is reading or writing a parameter to the cache, or is configuring the PLL. While the busy signal is asserted, no parameters can be read or written, and no reconfiguration can be initiated. Changes to the megafunction can be made only when the busy signal is not asserted. The signal goes high when the read_param, write_param, or reconfig input port is asserted, and remains high until the specified operation is complete. In the case of a reconfiguration operation, the busy signal remains high until the pll_areset signal is asserted and then deasserted.
pll_areset	Yes	Drives the areset port on the PLL to be reconfigured. The pll_areset port must be connected to the areset port of the ALTPLL megafunction for the reconfiguration to function correctly. This signal is active high. pll_areset is asserted when pll_areset_in is asserted, or, after reconfiguration, at the next rising clock edge after the scandone signal goes high. If you use the ALTPLL_RECONFIG megafunction, drive the PLL areset port using the pll_areset output port.
pll_configupdate	No	Drives the configupdate port on the PLL to be reconfigured. When asserted, the pll_configupdate port loads selected data to PLL configuration latches. The signal is asserted after the final data bit is sent out. This port is available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices only.
pll_scanclk	Yes	Drives the scanclk port on the PLL to be reconfigured. For information about the maximum scanclk frequency for the various devices, refer to the respective device handbook.
pll_scanclkena	No	This acts as a clock enable for the scanclk port on the PLL to be reconfigured. Reconfiguration begins on the first rising edge of pll_scanclk after pll_scanclkena is asserted. On the first falling edge of pll_scanclk, after the pll_scanclkena signal is deasserted, the megafunction stops scanning data to the PLL.
pll_scanaclr	Yes	Drives the scanaclr port on the PLL to be reconfigured. Not available for Stratix II, Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy II, HardCopy III, HardCopy IV, Arria GX, and Arria II GX devices.
pll_scanread	No	Drives the scanread port on the PLL to be reconfigured. Not available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices.
pll_scanwrite	No	Drives the scanwrite port on the PLL to be reconfigured. Not available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices.

**Table 9. ALTPLL\_RECONFIG Megafunction Output Ports (Part 2 of 2)**

Port Name	Required?	Description
pll_scandata	Yes	Drives the scandata port on the PLL to be reconfigured. This output port from the megafunction holds the scan data input to the PLL for the dynamically reconfigurable bits. The pll_scandata port sends scandata to the PLL. Any activity on this port can only be observed when the reconfig signal is asserted.
rom_address_out	No	Specifies the address in ROM from which data is written to the scan chain. During the write operation, the address increments from address 0 to the size of the scan cache. The rom_address_out port is valid only while the write_rom_ena port is asserted. Each address is asserted for 1 clock cycle. This port is available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices only.
write_rom_ena	No	Enables the ROM. When the write_rom_ena signal is asserted, the ALTPLL_RECONFIG megafunctions generates valid addresses on the rom_address_out port. This port is available for Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices only.

Table 10 lists the ALTPLL\_RECONFIG megafunction parameters.

**Table 10. ALTPLL\_RECONFIG Megafunction Parameters (Part 1 of 2)**

Parameter	Type	Required?	Description						
init_from_rom[]	String	No	Specifies initialization from ROM. Initializes on power-up and not after every reset (to match RAM .mif file behavior). The available values are YES and NO. If omitted, the default value is NO.						
pll_type	String	No	Specifies the type of PLL to instantiate. For Stratix III, Stratix IV, Cyclone III, Cyclone IV, Arria II GX, HardCopy III, and HardCopy IV devices, values are TOP_BOTTOM and LEFT_RIGHT. For Stratix II, Stratix II GX, Arria GX, and HardCopy II devices, values are ENHANCED and FAST.						
scan_chain	String	No	Specifies the valid PLL types. Available for Stratix and Stratix GX devices only. Values are LONG, SHORT, and UNUSED. If omitted, the default value is UNUSED.						
			<table><thead><tr><th>Value</th><th>Configuration Setting</th></tr></thead><tbody><tr><td>LONG</td><td>288-bit scan chain used to control PLLs with 12 counters</td></tr><tr><td>SHORT</td><td>192-bit scan chain used to control PLLs with 8 counters</td></tr></tbody></table>	Value	Configuration Setting	LONG	288-bit scan chain used to control PLLs with 12 counters	SHORT	192-bit scan chain used to control PLLs with 8 counters
			Value	Configuration Setting					
LONG	288-bit scan chain used to control PLLs with 12 counters								
SHORT	192-bit scan chain used to control PLLs with 8 counters								

**Table 10. ALTPLL\_RECONFIG Megafunction Parameters (Part 2 of 2)**

Parameter	Type	Required?	Description												
scan_init_file	String	No	<p>Specifies the name of the <b>.mif</b> or <b>.hex</b> used as the initial value of the scan chain cache. Values are <i>&lt;filename&gt;</i> and <b>UNUSED</b>. If omitted, the value for every entry in the cache is 0. You must set the input file as shown in the following table:</p> <table><thead><tr><th>Device</th><th>Input File Size</th></tr></thead><tbody><tr><td>Stratix series devices (except Stratix III and Stratix IV devices)</td><td>192 bits or 288 bits depending on the scan_chain length</td></tr><tr><td>Stratix III and Stratix IV Top/Bottom PLL devices</td><td>234 bits</td></tr><tr><td>Stratix III and Stratix IV Right/Left PLL devices</td><td>180 bits</td></tr><tr><td>Cyclone III and Cyclone IV devices</td><td>144 bits</td></tr><tr><td>Arria II GX devices</td><td>180 bits</td></tr></tbody></table>	Device	Input File Size	Stratix series devices (except Stratix III and Stratix IV devices)	192 bits or 288 bits depending on the scan_chain length	Stratix III and Stratix IV Top/Bottom PLL devices	234 bits	Stratix III and Stratix IV Right/Left PLL devices	180 bits	Cyclone III and Cyclone IV devices	144 bits	Arria II GX devices	180 bits
Device	Input File Size														
Stratix series devices (except Stratix III and Stratix IV devices)	192 bits or 288 bits depending on the scan_chain length														
Stratix III and Stratix IV Top/Bottom PLL devices	234 bits														
Stratix III and Stratix IV Right/Left PLL devices	180 bits														
Cyclone III and Cyclone IV devices	144 bits														
Arria II GX devices	180 bits														
use_scanclk_sync_register	String	No	<p>Specifies whether to use the scanclk port for the synchronization mode for the register. Available for all supported devices except Stratix III, Stratix IV, Cyclone III, Cyclone IV, HardCopy III, HardCopy IV, and Arria II GX devices. Values are <b>YES</b> and <b>NO</b>. If omitted, the default value is <b>NO</b>.</p>												

**Table 11** lists the counter\_type settings for Stratix III, Stratix IV, Cyclone III, Cyclone IV, and Arria II GX devices.

**Table 11. counter\_type[3..0] Settings for Stratix III, Stratix IV, Cyclone III, Cyclone IV, and Arria II GX Devices (Part 1 of 2)**

Counter Selection	Binary	Decimal
N	0000	0
M	0001	1
CP/LF	0010	2
VCO	0011	3
C0 (1)	0100	4
C1 (1)	0101	5
C2 (1)	0110	6
C3 (1)	0111	7
C4 (1)	1000	8
C5 (2)	1001	9
C6 (2)	1010	10
C7 (3)	1011	11
C8 (3)	1100	12
C9 (3)	1101	13
Illegal value	1110	14

**Table 11. counter\_type[3..0] Settings for Stratix III, Stratix IV, Cyclone III, Cyclone IV, and Arria II GX Devices (Part 2 of 2)**

Counter Selection	Binary	Decimal
Illegal value	1111	15

**Notes to Table 11:**

- (1) For Stratix III and Stratix IV Top/Bottom PLL, Stratix III and Stratix IV Left/Right PLL, and Cyclone III and Cyclone IV PLL.
- (2) For Stratix III and Stratix IV Top/Bottom PLL, Stratix III and Stratix IV Left/Right PLL, and Arria II GX PLL.
- (3) For Stratix III and Stratix IV Top/Bottom PLL only.

Table 12 lists the counter\_param settings for Stratix III, Stratix IV, Cyclone III, Cyclone IV, and Arria II GX devices.

**Table 12. counter\_param[2..0] Settings for Stratix III, Stratix IV, Cyclone III, Cyclone IV, and Arria II GX Devices**

Counter Type	Counter Param	Binary	Decimal	Width (bits)
Regular counters C0-C9: Top-Bottom Stratix III and Stratix IV C0-C6: Left-Right Stratix III and Stratix IV C0-C4: Cyclone III and Cyclone IV C0-C6: Arria II GX	High count	000	0	8
	Low count	001	1	8
	Bypass	100	4	1
	Mode (odd/even division)	101	5	1
CP/LF	Charge pump unused	101	5	5
	Charge pump current	000	0	3
	Loop filter unused	100	4	1
	Loop filter resistor	001	1	5
	Loop filter capacitance	010	2	2
VCO	VCO Post Scale	000	0	1
M/N Counters	High count	000	0	8
	Low count	001	1	8
	Bypass	100	4	1
	Mode (odd/even division)	101	5	1
	Nominal count	111	7	9

**Note to Table 11:**

- (1) For even nominal count, the counter bits are automatically set as follows:

- $high\_count = Nominalcount/2$
- $low\_count = Nominalcount/2$

For odd nominal count, the counter bits are automatically set as follows:

- $high\_count = (Nominalcount + 1)/2$
- $low\_count = Nominalcount - high\_count$
- odd/even division bit = 1

For nominal count = 1, Bypass bit = 1.

Table 13 lists the `counter_type` and `counter_param` settings for Stratix II, Stratix II GX, Arria GX, and HardCopy II devices.

**Table 13. `counter_type` and `counter_param` Settings for Stratix II, Stratix II GX, Arria GX, and HardCopy II Devices (Part 1 of 2)**

Counter	Settings	
<code>counter_type[ ]</code>	4-bit bus that selects which counter type must be updated. The following mapping determines which counter is specified for each <code>counter_type</code> value.	
	<code>counter_type[3..0]</code>	Selected Counter
	0000 (0x0)	N [Enhanced PLL/ Fast PLL]
	0001 (0x1)	M [Enhanced PLL/ Fast PLL]
	0010 (0x2)	P/LF [Enhanced PLL/ Fast PLL]
	0011 (0x3)	(Illegal value)
	0100 (0x4)	C0 [Enhanced PLL/ Fast PLL]
	0101 (0x5)	C1 [Enhanced PLL/ Fast PLL]
	0110 (0x6)	C2 [Enhanced PLL/ Fast PLL]
	0111 (0x7)	C3 [Enhanced PLL/ Fast PLL]
	1000 (0x8)	C4 [Enhanced PLL]
	1001 (0x9)	C5 [Enhanced PLL]
	1010 (0xA)	(Illegal value)
	1011 (0xB)	(Illegal value)
	1100 (0xC)	(Illegal value)
	1101 (0xD)	(Illegal value)
	1110 (0xE)	(Illegal value)
	1111 (0xF)	(Illegal value)



**Table 13. counter\_type and counter\_param Settings for Stratix II, Stratix II GX, Arria GX, and HardCopy II Devices (Part 2 of 2)**

Counter	Settings		
counter_param[ ]	3-bit bus that selects which parameter for the given counter type should be updated. The mapping to each parameter type and the corresponding parameter bit-width is defined as follows:		
	counter_param[2..0]	Selected Parameter	Width
	000 (0x0)	nominal count <sup>(1)</sup>	9
	001 (0x1)	spread count <sup>(1)</sup>	9
	000 (0x0)	high cycles count <sup>(2)</sup>	8
	001 (0x1)	low cycles count <sup>(2)</sup>	8
	000 (0x0)	high cycles count <sup>(3)</sup>	4
	001 (0x1)	low cycles count <sup>(3)</sup>	4
	000 (0x0)	nominal count <sup>(4)</sup>	2
	010 (0x2)	phase step setting	2
	100 (0x4)	counter bypass bit	1
	101 (0x5)	counter odd division bit <sup>(5)</sup>	1
	101 (0x5)	spread counter bypass <sup>(1)</sup>	1
	000 (0x0)	charge pump current <sup>(6)</sup>	4
	001 (0x1)	loop filter resistor <sup>(6)</sup>	6
	010 (0x2)	loop filter capacitor <sup>(6)</sup>	2
	011 (0x3)	(illegal value)	—
	110 (0x6)	(illegal value)	—
	111 (0x7)	(illegal value)	—

**Notes to Table 13:**

- (1) For Enhanced PLL M and N.
- (2) For Enhanced PLL C0-C5.
- (3) For Fast PLL C0-C3 and M.
- (4) For Fast PLL N.
- (5) For C0-C5, Fast PLL M.
- (6) For CP/LF.

Table 14 lists the `counter_type` and `counter_param` settings for Stratix and Stratix GX devices.

**Table 14. `counter_type` and `counter_param` Settings for Stratix and Stratix GX Devices (Part 1 of 2)**

Counter	Settings	
<code>counter_type[ ]</code>	4-bit bus that selects which counter type must be updated. The following mapping determines which counter is specified for each <code>counter_type</code> value.	
	<code>counter_type[3..0]</code>	Selected Counter <sup>(1)</sup>
	0000 (0x0)	M
	0001 (0x1)	N
	0010 (0x2)	(Illegal value)
	0011 (0x3)	(Illegal value)
	0100 (0x4)	G0
	0101 (0x5)	G1
	0110 (0x6)	G2
	0111 (0x7)	G3
	1000 (0x8)	L0
	1001 (0x9)	L1
	1010 (0xA)	(Illegal value)
	1011 (0xB)	(Illegal value)
	1100 (0xC)	E1
	1101 (0xD)	E2
	1110 (0xE)	E3
	1111 (0xF)	E4

**Table 14. counter\_type and counter\_param Settings for Stratix and Stratix GX Devices (Part 2 of 2)**

Counter	Settings		
counter_param[ ]	3-bit bus that selects which parameter for the given counter type should be updated. The mapping to each parameter type and the corresponding parameter bit-width is defined as follows:		
	counter_param[ 2..0 ] <sup>(5)</sup>	Selected Parameter	Width
	000 (0x0)	nominal count <sup>(2)</sup>	9
	001 (0x1)	spread count <sup>(2)</sup>	9
	000 (0x0)	high cycles count <sup>(3)</sup>	9
	001 (0x1)	low cycles count <sup>(3)</sup>	9
	010 (0x2)	delay element setting	4
	011 (0x3)	(illegal value)	—
	100 (0x4)	counter bypass bit	1
	101 (0x5)	counter odd division bit <sup>(4)</sup>	1
	110 (0x6)	(illegal value)	—
	111 (0x7)	(illegal value)	—

**Notes to Table 14:**

- (1) The lower 2 bits indicate the counter number for the G, L, and E counters. Additionally, the E<sub>x</sub> counters are only valid for the LONG scan chain PLLs.
- (2) For M and N.
- (3) For G, L, and E.
- (4) Must be 0 for M or N.
- (5) The upper 2 bits determine the width (for example, 00<sub>x</sub> = 9, 01<sub>x</sub> = 4, 10<sub>x</sub> = 1)

## Document Revision History

Table 15 lists the revision history for this document.

**Table 15. Document Revision History (Part 1 of 2)**

Date	Version	Changes
February 2012	6.0	<ul style="list-style-type: none"> <li>Added information about nominal count usage in Table 12.</li> <li>Updated the decimal value in Table 12.</li> </ul>
August 2010	5.0	Updated for the Quartus II software version 10.0.
July 2008	4.0	<ul style="list-style-type: none"> <li>Added support for Stratix IV devices.</li> <li>Updated screenshots for “MegaWizard Plug-In Manager Page Descriptions” section.</li> <li>Updated “Features” section.</li> <li>Updated “General Description” section.</li> <li>Added “Simulation” section.</li> <li>Added “Checking Design Violations With the Design Assistant” section</li> <li>Added comment for pll_scanclock port.</li> <li>Added support for new port, pll_ena.</li> </ul>

**Table 15. Document Revision History (Part 2 of 2)**

Date	Version	Changes
November 2007	3.2	<ul style="list-style-type: none"> <li>■ Updated for the Quartus II software version 7.2, including: Added new user-mode reconfiguration from external ROM.</li> <li>■ Added three new input ports: rom_data_in, write_from_rom, and reset_rom_address</li> <li>■ Added two new output ports: rom_address_out and write_rom_ena</li> <li>■ Updated design example format</li> <li>■ Updated: <a href="#">Figure 5</a>, <a href="#">Figure 6</a>, <a href="#">Figure 7</a>, <a href="#">Figure 17</a>, <a href="#">Figure 18</a>, <a href="#">Figure 19</a>, <a href="#">Figure 20</a>, <a href="#">Figure 21</a>, <a href="#">Figure 22</a></li> <li>■ Changed altpII_reconfig to ALTPLL_RECONFIG throughout the document.</li> </ul>
March 2007	3.1	Added Cyclone III to list of supported devices.
December 2006	3.0	Updated for Quartus II software version 6.1.
October 2006	2.0	Updated for Quartus II software version 6.0.
April 2005	1.0	Initial release.